Structuring Multi-Layer Scalable Streams to Maximize Client-Perceived Quality

Cheng-Hsin Hsu
School of Computing Science
Simon Fraser University
Surrey, BC, Canada
cha16@cs.sfu.ca

Mohamed Hefeeda
School of Computing Science
Simon Fraser University
Surrey, BC, Canada
mhefeeda@cs.sfu.ca

***Abstract*— Recent video coders, such as H.264/SVC, can encode a video stream into multiple layers, each with a different rate. Moreover, each layer can either be coarse-grained scalable (CGS) or fine-grained scalable (FGS). FGS layers support wider ranges of client bandwidth than CGS layers, but suffer from higher coding inefficiency. Currently there are no systematic ways in the literature to determine the optimal stream structure that renders the best average quality for all clients. In this paper, we formulate an optimization problem to determine the optimal rate and encoding granularity (CGS or FGS) of each layer in a scalable video stream that maximizes a system-defined utility function for a given client distribution. We design an efficient, yet optimal, algorithm to solve this optimization problem. Our algorithm is general in the sense that it can employ arbitrary utility functions for clients. We implement our algorithm and verify its optimality. We show how various structuring of scalable video streams affect individual client utilities. We compare our algorithm against a heuristic algorithm that has been used before in the literature, and we show that our algorithm outperforms the other one in all cases.**

I. Introduction

Video streaming over the Internet is increasingly getting very popular as higher bandwidth links and more powerful machines are becoming more affordable for end users. Users typically seek the highest possible video quality. Users, however, are quite heterogeneous in terms of network bandwidth and processing capacity. A conventional nonscalable coded stream supports only one decoding rate, which is insufficient in such a heterogeneous environment. This is because supporting clients with different bandwidth requires storing and serving multiple versions of each video stream. To cope with this heterogeneity, various scalable coding techniques have been proposed in the literature. A scalable coded stream consists of various representations of the original video sequence, with different resolutions, frame rates, or quality levels.

Scalable coders compress video data into a base layer that provides basic quality, and multiple enhancement layers that add incremental quality refinements. Current video coders, e.g., H.264/SVC [1], allow the enhancement layers to be either coarse-grained scalable or fine-grained scalable. Fig. 1 shows the general structure of a scalable video stream that can be produced by the H.264 reference software [2]. Coarse-grained scalable (CGS) layers provide limited rate scalability: clients receiving incomplete CGS layers cannot use them to enhance quality. In contrast, fine-grained scalable (FGS) layers provide quality refinements proportional to the number of *bits* received [1], [3], [4]. FGS layers, thus, support wider ranges of client bandwidth and it can fully utilize available bandwidth of individual clients, which results in better video playback quality and ultimately higher user satisfaction. The fine rate scalability of FGS, however, comes at an expense of coding efficiency. That is, an FGS coded layer results in lower quality compared to a CGS coded layer at the same bit rate [1], [5]. Hence, there is a trade-off between coding efficiency and supported rate range, which can be gauged by the scalable stream structure. More importantly, the average perceived quality for all clients depends on this trade-off, and thus depends on the choice of stream structure.

Although modern video coders can encode a video stream into multiple CGS and FGS layers, currently there are no systematic ways in the literature to determine the optimal stream structure that renders the best average quality for all clients.

In this paper, we address the problem of structuring scalable video streams. We formulate an optimization problem to determine the optimal rate and encoding granularity (CGS or FGS) of each layer in a scalable video stream that maximizes a system-defined utility function for a given client distribution. We design an efficient, yet optimal, algorithm to solve this optimization problem. Our algorithm is general in the sense that it can employ arbitrary utility functions for clients. We implement our algorithm , and show how various structuring of scalable video streams affect the client utilities. To demonstrate the generality of our algorithm, we consider three utility functions in our experiments, which model various aspects including effective rate received by clients, the mismatch between client bandwidth and received stream rate, and client perceived quality in terms of PSNR. We also compare our algorithm against another algorithm that has been used before in the literature, and we show that our algorithm outperforms the other one in all cases.

The rest of this paper is organized as follows. In Section II, we summarize the related works. In Section III, we discuss and model the overhead associated with scalable streams. Then, we formulate the optimization problem, and present our algorithm to solve it. We evaluate our algorithm in Section IV, and we

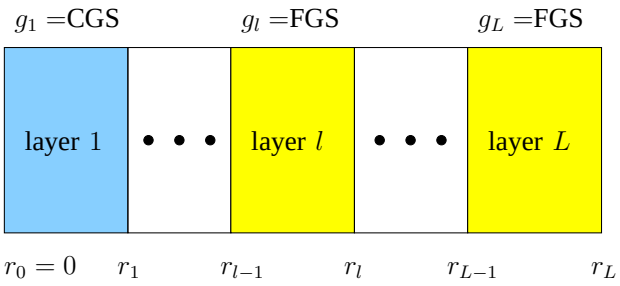


Fig. 1. General structuring of a scalable video stream with L layers. Each layer l has a coding rate r_l and a scalability type g_l which can be either coarse grain (CGS) or fine grain (FGS). This structure can be produced by H.264/SVC coders.

conclude the paper in Section V.

II. Related Work

Streaming systems [6]–[9] account for the coding efficiency using a layering overhead function, which represents the bit rate that does not contribute toward the video quality. Similarly, in our formulation we use an overhead function that depends on the rate of the layer being coded as well as the cumulative rate of preceding layers.

The performance of multi-layer versus multi-version streams is studied in [10]. The authors formulate an optimization problem to compute the rate of each layer such that the average perceived video quality is maximized. The square root rate-distortion model [11] is used to estimate the coding efficiency of the layered coding. In [9], the authors consider broadcasting multi-layer video streams in a wireless cellular system with a given number of channels and client capacity distribution. They determine the optimal rate of each layer to maximize the average perceived quality. Unlike our work, these works target coarse grained scalable video streams, and do not consider fine grained scalable streams.

The authors of [12] study multicast streaming systems with many receivers. They partition receivers into several groups to maximize a system-wide utility function. Their algorithm uses a general utility function $u(r_c, b_c)$ of the bandwidth b_c and the streaming rate r_c. Two utility functions are employed in their experiments: (i) $\min(r_c, b_c)$ as the received rate, and (ii) $\min(r_c, b_c)/\max(r_c, b_c)$ as the inter-receiver-fairness index. We use similar utility functions in our work. A video stream used in such systems can be encoded into multiple layers. Multiple versions with different rates of the same stream can also be created. This work does not consider fine-grained streams, nor does it account for the layering overhead.

Finally, in our previous work [13], we considered structuring MPEG-4 streams which can have one base layer and one fine-grained enhancement layer. The problem in [13] was to compute the optimal width of the base layer. In the current paper, we consider multiple-layer streams and each layer can have different scalability granularity.

III. Problem Formulation and Solution

In this section, we discuss and model the overhead associated with scalable streams. Then, we formulate the optimization problem, and present our algorithm to solve it.

A. Modeling Scalability Overhead

We consider scalable streams that can be structured into L layers, as shown in Fig. 1. Compared to nonscalable coders, a scalable coder imposes more overhead on streaming systems. This overhead includes reduced compression efficiency, and added protocol headers. We collectively call these overheads as the *scalability overhead*. We capture the effect of the scalability overhead by using an overhead function a and the effective rate $\bar{r}$ notion, which is formalized in the following definition.

Definition 1 (Effective Rate of a Scalabe Stream): Consider a scalable stream encoded at rate r. The effective rate $\bar{r}$ of that stream is equal to the rate of the nonscalable stream that produces the same quality. Furthermore, $\bar{r}$ is given by $r/(1+a)$, where a is a function that accounts for the scalability overhead.

In the above definition, the a function specifies the fraction of the total stream rate that does not contribute to the video playback quality. Defining the effective rate in this way enables us to compare various scalability methods, i.e., CGS and FGS, against each other and against nonscalable encoding.

The scalability overhead function is an input to our stream structuring algorithm, and it can be estimated using either experimental or analytical methods. Some guidelines on estimating this function are in order though. In general, the scalability overhead function a depends on three factors: (i) characteristics of the video sequence, (ii) granularity of the scalable coding, and (iii) rate of the layer being encoded as well as the rates of its preceding layers. We discuss each of these factors in the following. First, the experimental study in [5] indicates that video sequences with more temporal redundancy incur higher scalability overhead. In addition, video sequences with similar amount of temporal redundancy have similar scalability overheads. This suggests categorizing video sequences based on temporal redundancy and computing an overhead function for each category. Second, as indicated by previous studies [1], [5], fine-grained scalable coding imposes more overhead than coarse-grained scalable coding. To model this difference, we use two overhead functions: a_0 and a_1 for CGS and FGS layers, respectively. Finally, the authors of [5] observe that encoding the base layer of MPEG-4 FGS sequences at higher rates yields lower scalability overhead for the enhancement layer. This indicates that the overhead function of a layer will depend on the cumulative rates of the preceding layers, in addition to the rate of that layer itself. To model this dependence, we define the effective rate $\bar{r}_l$ of layer l $(1 \le l \le L)$ as follows:

$$\bar{r}_l = \begin{cases} r_1, & l = 1 \\ \bar{r}_{l-1} + \frac{r_l - r_{l-1}}{1+a(r_l)}, & 2 \le l \le L \end{cases} \quad (1)$$

In the above equation, we use r_l to denote the encoding rate of layer l. Layer 1 (base layer) does not incur scalability overhead (i.e., $\bar{r}_1 = r_1$), because it is typically encoded using a nonscalable method. For successive (enhancement) layers, the effective rate of layer l is computed recursively from the effective rate of layer $l-1$ and the width of layer l scaled down by the overhead function $a(r_l)$. We scale down the width of layer l to account for the scalability overhead. We use the effective rate defined in (1) in our problem formulation.

B. Problem Formulation

Our goal in this paper is to find the optimal structure of a multi-layer scalable video stream. That is, we want to compute the coding method (CGS or FGS) and the coding rate of each layer to maximize a system-wide utility function. We elaborate on the utility function later in this section.

We consider heterogeneous client populations by dividing clients into C classes. All clients belonging to the same class c $(1 \leq c \leq C)$ have the same bandwidth b_c. We assume that $b_1 < b_2 < \cdots < b_C$ without loss of generality. The fraction of clients in each class c is given by a probability mass function $f(c)$, where $\sum_{c=1}^{C} f(c) = 1$. No assumptions are made on the number of client classes or on the probability function. Without loss of generality, we assume that $b_C \leq r_{max}$, which is a pre-determined maximum rate of the stream. If otherwise, we combine clients with bandwidth larger than r_{max} in a class with bandwidth equal to r_{max}. We can do that because no matter how large the client bandwidth is, it cannot receive more than the maximum rate r_{max}.

For client class c, its actual received rate is no larger than b_c. To account for scalability overhead, we define $\bar{b}_c$ to be the effective rate of client class c, where $1 \leq c \leq C$. $\bar{b}_c$ is a function of the adopted structuring policy, which is defined as $S = \{(r_i, g_i), i = 1, 2, \ldots, L\}$, where r_i determines the encoding rate, and g_i decides the granularity for layer i. We set $g_i = 0$ if layer i is CGS-coded, and $g_i = 1$ if it is FGS-coded. We assume $g_1 = 0$, because the base (first) layer is typically coded with nonscalable coders, which do not incur scalability overhead. We use l to denote the highest layer that can be transferred to client c in its entirety (i.e., $r_l \leq b_c \leq r_{l+1}$), we write the effective rate $\bar{b}_c$ as:

$$\bar{b}_c = \begin{cases} \bar{r}_l, & g_i = 0 \\ \bar{r}_l + \frac{b_c - r_l}{1 + a_1(r_{l+1})}, & g_i = 1 \end{cases} \tag{2}$$

The effective rate of class c is equal to that of layer l, if layer $l+1$ is CGS-coded. If layer $l+1$ is FGS-coded, the additional rate $b_c - r_l$ can be received on top of $\bar{r}_l$, which contributes to the effective rate of class c after being scaled down by the FGS overhead function $a_1(r_{l+1})$.

Our problem can formally be stated as follows. Given a scalable stream that can be structured into up to L layers, and a large number of clients divided into C classes with their distribution given by the probability mass function $f(c)$, find the optimal structuring policy $S^* = \{(r_i^*, g_i^*), i = 1, 2, \ldots, L\}$ that yields the maximum system-wide utility Y_0^*, which is defined as the average client utility over all classes. Mathematically, we write our problem as:

$$P_0 : Y_0^* = \max_S \sum_{k=1}^{C} f(k) u(\bar{b}_k, b_k) \tag{3a}$$

$$\text{s.t.} \quad r_1 < r_2 < \cdots < r_L; \tag{3b}$$

$$g_1 = 0; \tag{3c}$$

$$g_i \in \{0, 1\}, \ \forall i = 2, 3, \ldots, L. \tag{3d}$$

In the above formulation, the utility function $u(\bar{b}_c, b_c)$ is a non-decreasing function of the effective rate achieved by client c. We use the effective rate in the utility function to account for the scalability overhead. We do not impose any restrictions on the utility function: It can be any arbitrary function that may, for example, describe utilization of system resources, satisfaction of clients, or a combination of both. Our algorithm, presented in the next section, works with any user-defined utility function. In the evaluation section, we use three types of utility functions. These utility functions have been used before in the literature, and they model various aspects such as the effective rate received by clients, the mismatch between client bandwidth and received stream rate, and client perceived quality in terms of PSNR.

The optimization problem in (3) has an exponential number of feasible solutions, and exhaustively trying all of them to find the optimal one is extremely expensive. In the next subsection, we propose an efficient, yet optimal, algorithm to solve it. Our algorithm uses a dynamic programming approach.

C. Efficient Algorithm

We first develop two lemmas to reduce the search space size of our optimization problem P_0. We define a subproblem for (3) called $P(c, l)$, where $1 \leq c \leq C$, and $1 \leq l \leq L$. For this subproblem, we find the optimal structuring policy $S^* = \{(r_i^*, g_i^*), i = 1, 2, \ldots, l\}$ that yields the maximum system-wide utility $Y^*(c, l)$. We then solve this problem iteratively by utilizing solutions of smaller subproblems. In subproblem $P(c, l)$, we assume that the rate of layer 1 is higher than the bandwidth of client class $c-1$, and is no larger than the bandwidth of client class c. We also assume that the layer 1 is CGS-coded. Therefore, clients in class $c-1$ and below receive nothing and contribute zero system utility. We can write the subproblem $P(c, l)$ as:

$$Y^*(c, l) = \max_S \sum_{k=c}^{C} f(k) u(\bar{b}_k, b_k) \tag{4a}$$

$$\text{s.t.} \quad r_1 < r_2 < \cdots < r_l; \tag{4b}$$

$$g_1 = 0; \tag{4c}$$

$$g_i \in \{0, 1\}, \ \forall i = 2, 3, \ldots, l; \tag{4d}$$

$$b_{c-1} < r_1 \leq b_c. \tag{4e}$$

In the above subproblem, constraint (4e) enables us to reduce the search space. We incrementally relax this limitation to derive the optimal solution for the original problem P_0. Solving subproblem $P(c, l)$ is still hard. For instance, there are

too many possible solutions to consider in order to determine the optimal coding rate for layer 1 alone. We present the following lemma to reduce the search space of the optimal structure policy. The proof is given in [14].

Lemma 1: There exists at least one optimal solution for the subproblem $P(c,l)$ that has following property: $b_{c-1} < r_1^* \leq b_c < r_2^*$.

This lemma states that if there is an optimal solution that has two or more layers between two adjacent classes b_{c-1} and b_c, we can find another optimal solution with only one layer between these two classes. Therefore, we do not need to allocate two layers between adjacent classes, which reduces the search space. The following lemma further reduces the search space. Its proof is given in [14].

Lemma 2: There exists at least one optimal solution for the subproblem $P(c,l)$ with layer 1 coded at rate b_c. That is, at least one optimal solution has the following property: $b_{c-1} < r_1^* = b_c$.

These two lemmas state that to determine r_1^* for an optimal solution of subproblem $P(c,l)$, we only need to consider $r_1 = b_c$. Next, we consider rates and granularity for other layers for an optimal solution of subproblem $P(c,l)$. We do so by recursively solving subproblem $P(i,l-1)$, where $c+1 \leq i \leq C-l+1$. That is, we sequentially solve subproblems with 1 layer, 2 layers, until L layers.

We solve a general subproblem $P(c,l)$, where $1 \leq c \leq C$ and $1 \leq l \leq L$ as follows. We first solve subproblem $P(c,1)$ for $1 \leq c \leq C$. Previous lemmas tell us that setting $r_1 = b_c$ in $P(c,1)$ leads to an optimal solution. As layer 1 is CGS-encoded, clients in class $c-1$ and below receive nothing, and clients in class c and above receive layer 1. Therefore, the optimal system utility $Y^*(c,1)$ can be easily computed.

We then solve subproblem $P(c,l)$, where $l > 1$ and $1 \leq c \leq C$. Again, following previous lemmas, we know that setting $r_1 = b_c$ leads to an optimal solution. To determine the rates and granularity of other layers, we consider optimal solutions for $P(i,l-1)$, where $c+1 \leq i \leq C-l+1$. We do not consider subproblems with $i > C-l+1$, because solutions of these problems have at least one bandwidth interval $(b_{c-1}, b_c]$ that contains rates for two or more layers. Previous lemmas tell us that considering stream structures with only one layer between any adjacent client classes is sufficient to find an optimal solution, which enables us to ignore these subproblems.

We can use the following formulation to solve subproblem $P(c,l)$ by utilizing optimal solutions for smaller subproblems:

$$Y^*(c,l) = \max_{\substack{c+1 \leq i \leq C-l+1, \\ g_2 \in \{1,0\}}} \{Y^*(i,l-1) - \textit{DIFF}(c,l,i)\}, \quad (5)$$

where $Y^*(c,l)$ represents the optimal system utility for subproblem $P(c,l)$, and $\textit{DIFF}(c,l,i)$ denotes the system utility difference caused by adding a CGS-coded layer at b_c to the optimal solution of subproblem $P(i,l-1)$. The details on derivations of $\textit{DIFF}(c,l,i)$ are given in [14]. This formulation finds the subproblem $P(i,l-1)$ and the granularity g_2 that maximize system-wide utility for subproblem $P(c,l)$. Notice that, g_2 represents the granularity of the lowest layer for subproblem $P(i,l-1)$. g_2 was assumed to be CGS-coded in subproblem $P(i,l-1)$, and its optimal setting is determined when solving subproblem $P(c,l)$ using (5). An important property of (5) is that we effectively consider all possible combinations of r_1 and g_2. Therefore, we can use dynamic programming technique to optimally solve subproblem $P(c,l)$ for its optimal structure. The following theorem shows how to construct an optimal solution for problem P_0 based on optimal solutions for subproblems $P(c,l)$. Due to space limitations, we give the proof in [14].

Theorem 1 (Optimality): Let $S^*(c,l)$ denote the optimal l-layer structure for subproblem $P(c,l)$ that achieves the maximal system utility $Y^*(c,l)$. The optimal structure S^* for the original problem P_0 is the one that achieves maximum system utility: $Y_0^* = \max_{1 \leq c \leq C-L+1}\{Y^*(c,L)\}$.

The above theorem illustrates that the optimal structure for problem P_0 can be derived by finding the maximal system-wide utility among all optimal solutions for subproblems $P(c,L)$, where $1 \leq c \leq C-L+1$, while the optimal solutions for subproblems $P(c,L)$ can be found by iteratively solving smaller subproblems. We present the details and pseudo code of our algorithm are given in [14].

The following theorem gives the time and space complexities of our algorithm. We prove it in [14].

Theorem 2 (Complexity): The time complexity of our algorithm is $O(C^3L)$ and its space complexity is $O(CL)$, where C is the number of client classes and L is the number of layers in the video stream.

IV. Evaluation

In this section, we first describe our experimental setup. We then demonstrate that our algorithm allows various utility functions and produces optimal stream structures. Next, we compare our algorithm with a widely used heuristic stream structuring algorithm. Only sample results are shown here due to space limitations. Interested readers are referred to the extended version of this paper [14] for the complete experimental results and a study on the impact of choosing different utility functions.

A. Setup

Our algorithm allows user-specified scalability overhead functions. Previous studies [1], [5] reveal that FGS coded layers results in higher scalability overhead compared to CGS coded layers. Therefore, we define $a_0(r_l)$ and $a_1(r_l)$ for CGS and FGS overhead function, where $a_0(r_l) \leq a_1(r_l)$ at any layer rate r_l. In our experiments, we let $a_0(0) = 5\%$ and $a_1(0) = 20\%$. We let both CGS and FGS overhead reach zero when $r_l \geq 5000$ kbps. That is, we have the following scalability overhead functions: $a_0(r_l) = \max\{0.05 - 0.00001r_l, 0\}$, and $a_1(r_l) = \max\{0.20 - 0.00004r_l, 0\}$.

Our algorithm works with any utility function $u(\bar{b}_c, b_c)$, where $\bar{b}_c$ is the effective rate of the received stream, and b_c is the available bandwidth of client c. Three utility functions are employed in our experiments: $u_{rate}(\bar{b}_c, b_c) = \bar{b}_c$, which assumes that the higher the effective rate that a client receives,

the more satisfied that client will be; (ii) $u_{utilization}(\bar{b}_c, b_c) = \bar{b}_c/b_c$, which tries to match the rate received by a client with its bandwidth; and (iii) u_{psnr} which maximizes the client-perceived quality (in PSNR) by using a rate-distortion (R-D) model to map the effective rate to perceived quality.

For u_{psnr}, we adopt a recent H.264/AVC R-D function which assumes that the transform coefficients are Cauchy distributed [15]. The R-D function is given as: $D = cR^{-\gamma}$, where the distortion D is in mean-square error (MSE) and rate R is in bits per pixel. The model parameters c and γ are sequence dependents. The authors of [15] show that this model is more accurate than Laplacian and Gaussian based R-D models. We use $u_{psnr}(\bar{b}_c, b_c) = -10\log_{10}[15.3787(0.1184\bar{b}_c)^{-4}]$ in our experiments with CIF video sequences. We note that this R-D model is proposed for nonscalable H.264/AVC coded streams. It is, however, applicable in our experiments because we convert actual rates to effective rates, which are equivalent to the rates of nonscalable stream.

We have implemented our algorithm in Java. We denote our algorithm by ScsOpt in plots. We use a commodity Linux desktop for our experiments. We consider $100,000$ clients with network bandwidth distributed according to four representative distributions. The first distribution is uniform between 35 and 3005 kbps. The second is a bi-model distribution that consists of two normally-distributed peaks with means at 250 kbps and 1000 kbps, and standard deviations of 25 and 100. This bi-model distribution is skewed to the right: 80% of client classes are from the normal distribution with mean 1000 kbps. The third is a bi-model distribution with the same setting, except that it is skewed to the left: 80% of client classes are from the normal distribution with mean 250 kbps. The fourth is a multi-model distribution with three normal distributions, which represents a typical client distribution in today's Internet: 50% of clients are equipped with dial-up connections, which have a normal distribution with mean 40 kbps and standard deviation of 25 kbps; 35% of clients use DSL services, where the average bandwidth is 1000 kbps with standard deviation of 100 kbps; and 15% of clients have high-speed connections with average bandwidth 2000 kbps and standard deviation of 200 kbps.

B. Optimal Stream Structuring

Our algorithm takes client bandwidth distribution as input. It produces a stream structure that results in the highest utility. As mentioned above, we use three different utility functions in our experiments. These utility functions lead to different optimal stream structures. Fig. 2 shows the optimal stream structuring policies computed by our algorithm for 5-layer scalable stream in scenario IV with different utility functions. We see that the resulted stream structure is influenced by the chosen utility function. Specifically, the following stream structures are determined to be the optimal stream structure for each of the utility functions: (i) u_{rate}: $\{(735, \text{CGS}), (805, \text{CGS}), (855, \text{CGS}), (885, \text{CGS}), (2745, \text{FGS})\}$, (ii) $u_{utilization}$: $\{(5, \text{CGS}), (15, \text{CGS}), (25, \text{CGS}), (35, \text{CGS}), (2745, \text{FGS})\}$, and (iii) u_{psnr}: $\{(35, \text{CGS}), (45, \text{CGS}), (95, \text{FGS}), (685, \text{CGS}), (2745, \text{FGS})\}$. Elements in each 2-tuple represent coding rate in kbps and granularity, respectively.

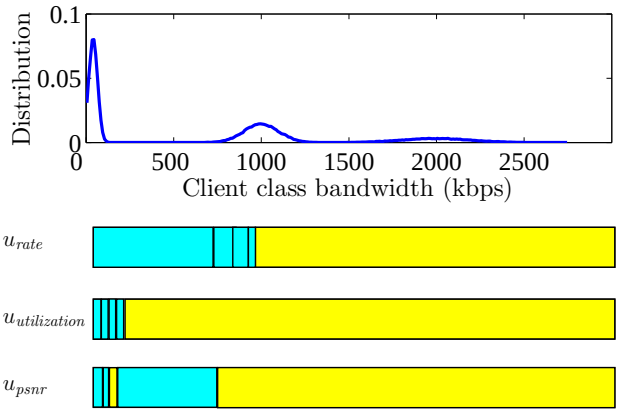


Fig. 2. Optimal structuring of a scalable video stream with 5 layers for scenario IV produced by our optimal algorithm with different utility functions.

These results show that our algorithm is general and can be used with various utility functions in different environments.

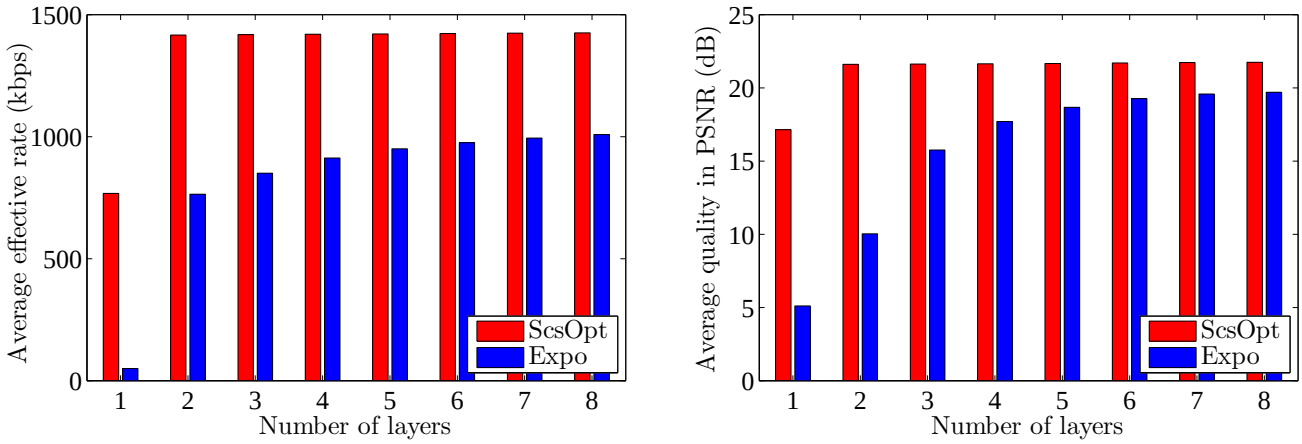
C. Comparison with Previous Structuring Algorithm

We compare the stream structures resulted by our algorithm against the heuristic structuring algorithm used in [9], [16]. This heuristic algorithm takes two rates for minimum and maximum supported decoding rates. It uses these two rates to code the first and the last layers, and then exponentially allocates rates for intermediate layers. That is, a layer l is assigned rate $r_{min}\rho^{l-1}$, where r_{min} and r_{max} are the minimum and maximum supported rates. The factor ρ is given by $\sqrt[L-1]{r_{max}/r_{min}}$, where L is the total number of layers. We use $r_{min} = 50$ kbps and $r_{max} = 1500$ kbps in our experiments. This covers a wide range of clients, from dial-up to broadband access links. We denote this algorithm by Expo in the plots. Fig. 3 illustrate the achieved system-wide utility by ScsOpt and Expo algorithms. Only sample results are shown due to the space limitation. More results are given in [14]. Our algorithm outperforms the heuristic algorithm with significant margins in all cases.

We note that Expo is the only algorithm we could find in the literature that may be applied to our stream structuring problem. Moreover, if there were other algorithms, the best they can do is to achieve results similar to our algorithm, because our algorithm is optimal as shown in Sec. III-C and experimentally verified in the extended version of this paper [14].

V. Conclusion

We have formulated an optimization problem to determine the optimal rate and encoding granularity (CGS or FGS) of each layer in a scalable video stream that maximizes a system-defined utility function for a given client distribution. We have



(a) Scenario I with u_{rate}

(b) Scenario IV with u_{psnr}

Fig. 3. Comparison between our algorithm (ScsOpt) and the heuristic algorithm (Expo) that exponentially allocates rates to layers. Sample data shown here due to space limitations.

proposed an optimal algorithm to solve this problem. Our algorithm is efficient and runs in $O(C^3L)$, where L is the number of layers in the video stream and C is the number of client classes. Since L and C are typically small integers, our algorithm is computationally efficient. Our algorithm can employ arbitrary utility functions for clients. To demonstrate the generality of our algorithm, we used three utility functions in our experimental study. These utility functions have been used before in the literature, and they model various performance metrics such as the effective rate received by clients, the mismatch between client bandwidth and received stream rate, and the client-perceived quality in terms of PSNR. We also compared our algorithm against another algorithm that has been used before in the literature, and we showed that our algorithm outperforms the other one in all cases.

In the extended version of this paper [14], we experimentally verified that our algorithm produces the optimal results and runs in a few seconds on a commodity PC. We studied the effect of various structuring of scalable video streams on client utilities for different utility functions. By analyzing various utility functions, we provided guidelines for content providers to choose the appropriate utility function that suits their needs.

ACKNOWLEDGMENT

This work is partially supported by the Natural Sciences and Engineering Research Council (NSERC) of Canada under Discovery Grant #313083 and RTI Grant #344619.

REFERENCES

[1] H. Schwarz, D. Marpe, and T. Wiegand, "The scalable H.264/MPEG4-AVC extension: Technology and applications," in *European Symposium on Mobile Media Delivery (EuMob'06)*, Sardinia, Italy, September 2006.

[2] Joint Video Team, "Joint scalable video model reference software," JSVM 8.0, February 2007.

[3] H. Radha, M. van der Schaar, and Y. Chen, "The MPEG-4 fine-grained scalable video coding method for multimedia streaming over IP," *IEEE Transactions on Multimedia*, vol. 3, no. 1, pp. 53–68, March 2001.

[4] W. Li, "Overview of fine granularity scalability in MPEG-4 video standard," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 11, no. 3, pp. 301–317, March 2001.

[5] M. van der Schaar and H. Radha, "Adaptive motion-compensation fine-granular-scalability (AMC-FGS) for wireless video," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 12, no. 6, pp. 32–51, June 2002.

[6] P. de Cuetos, D. Saparilla, and K. Ross, "Adaptive streaming of stored video in a TCP-friendly context: Multiple versions or multiple layers?" in *Proc. of International Packet Video Workshop (PV'01)*, Kyongju, Korea, April 2001.

[7] T. Kim and M. Ammar, "A comparison of layering and stream replication video multicast schemes," in *Proc. of ACM International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV'01)*, Port Jefferson, NY, June 2001, pp. 63–72.

[8] ——, "A comparison of heterogeneous video multicast schemes: layered encoding or stream replication," *IEEE Transactions on Multimedia*, vol. 7, no. 6, pp. 1123–1130, December 2005.

[9] J. Liu, B. Li, Y. Hou, and I. Chlamtac, "Dynamic layering and bandwidth allocation for multi-session video broadcasting with general utility functions," in *Proc. of IEEE INFOCOM'03*, San Francisco, CA, March 2003, pp. 630–640.

[10] I. Radulovic, P. Frossard, and O. Verscheure, "Adaptive video streaming in lossy networks: versions or layers?" in *Proc. of IEEE International Conference on Multimedia and Expo (ICME'04)*, Taipei, Taiwan, June 2004, pp. 1915–1918.

[11] M. Dai, D. Loguinov, and H. Radha, "Rate-distortion analysis and quality control in scalable Internet streaming," *IEEE Transactions on Multimedia*, vol. 8, no. 6, pp. 1135–1146, December 2006.

[12] Y. Yang, M. Kim, and S. Lam, "Optimal partitioning of multicast receivers," in *Proc. of IEEE International Conference on Network Protocols (ICNP'00)*, Osaka, Japan, November 2000, pp. 129–140.

[13] C. Hsu and M. Hefeeda, "Optimal partitioning of fine-grained scalable video streams," in *Proc. of ACM International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV'07)*, Urbana-Champaign, IL, June 2007.

[14] ——, "Optimal coding of multi-layer and multi-version video streams," Simon Fraser University, Tech. Rep. TR 2007-13, May 2007, available online at http://www.cs.sfu.ca/∼mhefeeda/publications.html.

[15] N. Kamaci, Y. Altunbasak, and R. Mersereau, "Frame bit allocation for the H.264/AVC video coder via Cauchy-density-based rate and distortion models," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 15, no. 8, pp. 994–1006, August 2005.

[16] J. Liu, B. Li, and Y. Zhang, "Optimal stream replication for video simulcasting," *IEEE Transactions on Multimedia*, vol. 8, no. 1, pp. 162–169, February 2006.