Optimal Coding of Multilayer and Multiversion Video Streams

Cheng-Hsin Hsu and Mohamed Hefeeda, *Member, IEEE*

Abstract—**Traditional video servers partially cope with heterogeneous client populations by maintaining a few versions of the same stream with different bit rates. More recent video servers leverage multilayer scalable coding techniques to customize the quality for individual clients. In both cases, heuristic, error-prone, techniques are currently used by administrators to determine either the rate of each stream version, or the granularity and rate of each layer in a multilayer scalable stream. In this paper, we propose an algorithm to determine the optimal rate and encoding granularity of each layer in a scalable video stream that maximizes a system-defined utility function for a given client distribution. The proposed algorithm can be used to compute the optimal rates of multiversion streams as well. Our algorithm is general in the sense that it can employ arbitrary utility functions for clients. We implement our algorithm and verify its optimality, and we 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. These utility functions model various aspects of streaming systems, including 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 compare our algorithm against a heuristic algorithm that has been used before in the literature, and we show that our algorithm outperforms it in all cases.**

Index Terms—**Multimedia communication, scalable coding, video quality optimization, video streaming.**

I. Introduction

CLIENTS in video streaming systems are, in general, heterogeneous in terms of network bandwidth and processing capacity. The heterogeneity comes from many sources, including different Internet access technologies used by clients, unequal network distances between the server and individual clients, and different screen resolutions and CPU speeds of the clients' machines. To partially cope with this heterogeneity, traditional video servers maintain a few versions of the same stream with different bit rates. The bit rate of each stream version is *heuristically* chosen by the administrators based on pre-assumed client bandwidth distribution.

More recent video servers employ scalable coding techniques to produce a single stream that can easily be customized to serve heterogeneous clients. These coding techniques compress video data into a base layer that provides basic quality, and multiple enhancement layers that add incremental quality refinements. Current scalable video coders, e.g., H.264/SVC [1], allow the enhancement layers to be either coarse-grained scalable (CGS) or fine-grained scalable (FGS). Fig. 1 shows the general structure of a scalable video stream that can be produced by the H.264/SVC reference software [2]. Because partial CGS layers cannot be decoded, CGS layers provide limited rate scalability. FGS layers, on the other hand, 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 a cost of lower coding efficiency: FGS layers yield lower quality compared to CGS layers coded at the same bit rate [1], [5]. Similar to the multiversion case, selecting the granularity of different layers and setting their encoding rates in scalable coding systems are currently done manually by the administrators based on rule-of-thumb, error-prone, techniques.

In this paper, we propose an algorithm 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. The proposed algorithm can be used to compute the optimal rates of multiversion streams as well. Our algorithm is general in the sense that it can employ arbitrary utility functions for clients. We implement our algorithm and verify its optimality, and we 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. These utility functions model various aspects of streaming systems, including 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 compare our algorithm against a heuristic algorithm that has been used before

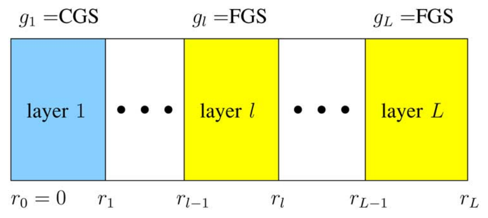

Fig. 1. General structuring of a scalable video stream with L layers. Each layer l has a coding rate r_l and a scalability type g_l which can be either coarse grain (CGS) or fine grain (FGS). This structure can be produced by H.264/SVC coders. Our proposed algorithm computes the optimal rate and granularity of each layer for a given client distribution.

Manuscript received May 1, 2007; revised August 23, 2007. This work was supported in part by the Natural Sciences and Engineering Research Council (NSERC) of Canada under Discovery Grant 313083 and RTI Grant 344619. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Marco Roccetti.

The authors are with the School of Computing Science, Simon Fraser University, Surrey, BC, V3T 0A3, Canada (e-mail: mhefeeda@cs.sfu.ca).

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.2007.911224

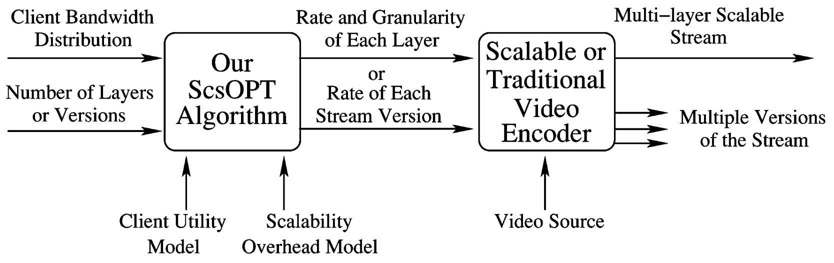
Client Bandwidth Distribution
Number of Layers or Versions
Our ScsOPT Algorithm
Rate and Granularity of Each Layer
or
Rate of Each Stream Version
Scalable or Traditional Video Encoder
Multi–layer Scalable Stream
Multiple Versions of the Stream
Client Utility Model
Scalability Overhead Model
Video Source

Fig. 2. Relationship between our algorithm and the video encoder in video streaming systems.

in the literature, and we show that our algorithm outperforms it in all cases.

The rest of this paper is organized as follows. In Section II, we discuss various applications of our algorithm in realistic environments. In Section III, we summarize the related works. In Section IV, we discuss and model the overhead associated with scalable streams. Then, we formulate an optimization problem to determine the optimal encoding rates and granularities of different layers in scalable streams. We also present an efficient algorithm to solve this optimization problem. We evaluate the proposed algorithm in Section V, and we conclude the paper in Section VI.

II. Motivations and Applications

Our proposed algorithm optimally solves the stream structuring problem in the order of seconds on a commodity PC. Searching for a good stream structures is not an easy task, as client distributions are heterogeneous and dynamic. Therefore, administrators of video servers may only find suboptimal stream structures using manual, rule-of-thumb, techniques, while our algorithm guarantees optimal stream structures.

Our algorithm assumes a fairly general model for stream structure (as shown in Fig. 1). Therefore, it can be used by streaming systems that employ various scalable as well as nonscalable video coders. Examples of such systems include the following.

- *H.264/SVC streams:* The emerging H.264/SVC standard aims to support highly heterogeneous clients over the Internet. It does so by providing flexible stream structures that enables multilayer coded stream, where each layer can be coded at a different rate with different granularity. Our algorithm produces the optimal structuring of these layers.
- *MPEG-4 FGS streams:* An MPEG-4 FGS stream consists of a nonscalable base layer and a single FGS enhancement layer. Streaming systems using such streams need to compute the rates of the base and enhancement layers. Our algorithm solves this problem by setting the number of layers to 2, and the granularity of the enhancement layer to FGS.
- *Traditional (MPEG-2) multilayer streams:* A traditional layered coded stream supports a few discrete decoding rates. Our algorithm solves the optimal structuring problem for these streams by fixing all layers to be CGS-encoded.
- *Multiversion nonscalable streams:* The widely-deployed multiversion streaming systems encode the same video into several versions at different rates. In such systems, administrators need to find the optimal rate for each version to achieve the best system performance. Our algorithm solves this problem as follows. We find the optimal structure of a stream with M CGS layers, where M is the desired number of versions. Then instead of actually encoding the stream into layers, we create M versions, each version $v (1 \leq v \leq M)$ is encoded at rate $\sum_{i=1}^{v} r_i$, where r_i is the rate of layer i.

Our algorithm (referred to as ScsOpt) can be placed in the big picture of video streaming systems as follows. The algorithm is to be implemented in a video server that serves either multilayer or multiversion streams. Fig. 2 shows the relationship between our algorithm and the video encoder. Our algorithm takes as input information about the bandwidth distribution of current clients as well as the number of desired layers (in case of scalable streams) or versions (in case of multiversion streams). It also takes into account the models describing the utility achieved by the clients and the overhead imposed by the scalable coding techniques. Our algorithm outputs the needed parameters for the video encoder to produce either a single multilayer scalable stream, or multiversions of the same stream but with different rates. Our algorithm can easily cope with the dynamic changes in client distributions, because it has short running time, and therefore, can re-compute the optimal stream structure for the updated client distribution with negligible overhead on the streaming server. For example, in a long live streaming session (e.g., sports events), the streaming server may collect statistics on clients during the session. Then, the server periodically (e.g., every 5 minutes) invokes our algorithm to compute the optimal stream structure for the current client distribution.

In addition, our algorithm can be used in unicast and multicast video streaming systems. In unicast, the server chooses the appropriate number of layers (or the closest version) for each client based on the client's capacity. In multicast with multilayer streams, the server transmits all layers through the distribution tree(s) and the clients obtain as many layers as their capacities allow. Depending on the details of the employed multicast protocol (e.g., IP or overlay multicast), not all layers will necessarily be transmitted through all branches of the tree(s). For multiversion streams, each version can have its own multicast session.

III. Related Work

To cope with heterogeneous client populations, multistream video systems have been proposed in the literature. In multistream systems, a receiver subscribes to one or a few streams that best-fit its bandwidth and processing power. This results in better client bandwidth utilization and higher video playback quality. Multistream systems can be classified into two categories based on their stream structures: i) multiversion systems that encode a video sequence into several independent streams at different rates and ii) multilayer systems that encode a video sequence into several nonoverlapped, dependent, streams. Readers are referred to [6] and [7] and references therein for a comprehensive list of multistream systems.

The destination set grouping (DSG) protocol is a representative multiversion streaming system [8], where a client subscribes to a stream that is coded at a rate no larger than its capability. In DSG, an intra-stream protocol is used to gauge stream rate within a pre-determined range, while an inter-stream protocol is employed to switch receivers among different stream versions. The receiver-driven layered multicast (RLM) is a representative multilayer streaming system [9], where a client subscribes to the base layer and a few enhancement layers, so that the total rate of these layers does not exceed its capability. Our proposed stream structuring algorithm is complementary to these multistream systems, because many of these systems, such as [8], [10], [11], concentrate on rate adaptation algorithms to minimize bandwidth mismatch by associating receivers with streams coded at given encoding rates. Our algorithm enables these systems to systematically find the optimal encoding rates (and granularity) that further minimize bandwidth mismatch once the rate adaptation algorithms converge. Hence, our algorithm improves bandwidth utilization of these multistream systems.

Optimal stream structuring problems that maximize system-wide video quality are considered in [12]–[16]. The authors of [12] formulate an optimization problem to compute transmission rates of individual stream versions that maximize a system-wide fairness utility function in a DSG-based multiversion streaming system. They propose a heuristic algorithm to solve this problem. In contrast, our algorithm is optimal and more general. A similar problem in multilayer streaming systems is studied in [13], where an optimal layering algorithm is proposed. However, the formulation does not model the layering overhead. The authors of [14] consider an optimization problem to find optimal rates for individual streams that maximize a general utility function $u(r_c, b_c)$, where b_c is client c's bandwidth and r_c is its streaming rate. Two utility functions are employed in their experiments: i) $\min(r_c, b_c)$, which models the bandwidth mismatch and ii) $\min(r_c, b_c)/\max(r_c, b_c)$, which models the inter-receiver fairness. We use similar utility functions in our work. Similar to our algorithm, the optimal rates produced by their algorithm can be used to encode a video stream into multiple layers, or multiple versions with different rates. The work in [14], however, does not consider fine-grained scalable streams, and ignores the coding inefficiency of scalable streams. In [15], the authors consider broadcasting multilayer 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. In [16], the authors consider the rate assignment problem in multiversion streaming systems. They determine an optimal rate for each stream version. Unlike our work, these works target coarse-grained scalable video streams, and do not consider fine-grained scalable streams.

Streaming systems, e.g., [15], [17]–[20] account for the coding inefficiency of scalable coders using a layering overhead function, which represents the bit rate that does not contribute toward the video quality. The authors of [17] uses the square root rate-distortion model [21] to approximate the layering overhead function. Several works assume a fixed layering overhead that is independent of stream structures [15], [18], [19]. The authors of [20] consider a dynamic layering overhead function that only depends on the base layer coding rate. Our formulation adopts a more elaborate scalability overhead function that depends on the rate of the layer being coded as well as the cumulative rate of preceding layers. More importantly, our formulation accounts for different coding granularity while previous works only consider coarse-grained scalable streams.

Finally, in our previous work [22], we considered structuring MPEG-4 streams which can have one base layer and one fine-grained enhancement layer. The problem in [22] 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.

IV. 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 additional 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 Scalable 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 function a 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:

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

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 multilayer 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 $(1 \le c \le C)$ 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 $b_C \le r_{\max}$, where $r_{\max}$ is a pre-determined maximum rate of the video stream. If otherwise, we combine all clients with bandwidth larger than $r_{\max}$ in a single 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 $1 \le c \le C$. $\bar{b}_c$ is a function of the adopted structuring policy S, 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., $r_l \le b_c \le r_{l+1}$), 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 $P_0(C, L)$

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

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

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

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

In the above formulation, the utility of a client is a nondecreasing function of the effective rate achieved by that client. 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 a few lemmas to reduce the search space of the optimization problem $P_0(C, L)$. We define a subproblem for (3) called $P(c, l)$, where $1 \le c \le C$, and $1 \le l \le L$. 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 this subproblem as $P(c,l)$ as follows:

$$Y^*(c,l) = \max_S \quad Y = \sum_{k=c}^{C} f(k)u(\bar{b}_k, b_k) \tag{4a}$$
$$\text{s.t.} \quad r_1 < r_2 < \cdots < r_l; \tag{4b}$$
$$g_1 = 0; \tag{4c}$$
$$g_i \in \{0,1\}, \quad i = 2,3,\ldots,l; \tag{4d}$$
$$b_{c-1} < r_1 \le b_c. \tag{4e}$$

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 $P_0(C,L)$. 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 two lemmas to reduce the search space of the optimal structure policy.

Lemma 1: For any given subproblem $P(c,l)$, there exists at least one optimal solution $S^* = \{(r_i^*, g_i^*), i = 1,2,\ldots l\}$ that has the following property: $b_{c-1} < r_1^* \le b_c < r_2^*$.

Proof: Let $S = \{(r_i, g_i), i = 1,2,\ldots,l\}$ be an optimal structuring policy of $P(c,l)$, with two or more layers coded at rates in $(b_{c-1}, b_c]$. Without loss of generality, we assume that there are two layers coded in this interval, i.e., $b_{c-1} < r_1 < r_2 \le b_c$. The cases with more than two layers coded in $(b_{c-1}, b_c]$ can be proved with the same technique. When S is employed, following (1), the clients in classes c and above receive the effective rate $r_1 + (r_2 - r_1)/(1 + a(r_2))$ from layers 1 and 2. We notice that, if we were to adopt a structuring policy $S^* = \{(r_i^*, g_i^*), i = 1,2,\ldots,l-1\}$, where $(r_i^*, g_i^*) = (r_{i+1}, g_{i+1})$, we still get optimal performance. This is because i) employing S^* allows clients in classes c and above to receive at the effective rate $r_1^* = r_2$, which is higher than the effective rate achieved by S as the overhead function is nonnegative and ii) clients in classes $c-1$ and below receive at the same effective rate regardless of whether S or S^* is employed. Thus, employing the structuring policy S^* achieves at least the same utility as S, because utility is a nondecreasing function. ■

Lemma 1 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. ■

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: $b_{c-1} < r_1^* = b_c$.

Proof: Let $S = \{(r_i, g_i), i = 1,2,\ldots,l\}$ be an optimal structuring policy of $P(c,l)$, where $b_{c-1} < r_1 < b_c$. We notice that setting $r_1 = b_c$ still produces an optimal structuring. This is because the utilities of clients in classes $c-1$ and below are not affected, and the clients in class c and above would achieve at least the same utility because utility is a nondecreasing function in terms of effective rate. ■

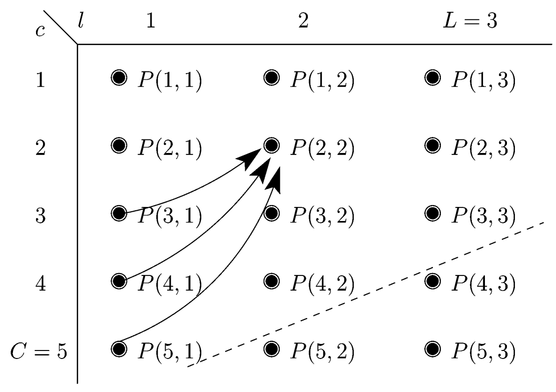

Fig. 3. Optimal structuring problem with C client classes and L layers can be solved using dynamic programming by dividing it into subproblems. Each problem is solved based on the results of preceding subproblems. Note that Lemma 1 enables us to skip subproblems in the lower right part.

These two lemmas imply 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 $c+1 \le i \le C-l+1$.

We present a simple example with three layers and five client classes to demonstrate the basic idea of our algorithm. The example is shown in Fig. 3. We take subproblem $P(2,2)$ as example, which finds the optimal 2-layer structuring policy, where all layers are coded at rates higher than b_1. From the previous lemmas, we know that setting $r_1 = b_2$ would result in an optimal solution. The subproblem $P(2,2)$ is then reduced to find the best coding rate and granularity for layer 2, which can be determined from subproblems $P(3,1), P(4,1)$, and $P(5,1)$. These three subproblems consider that the rate of layer 2 is in intervals $(b_2,b_3], (b_3,b_4]$, and $(b_4,b_5]$, respectively. This indeed covers all possible rates for layer 2 because we know that $r_1 < r_2$. To decide whether layer 2 should be FGS- or CGS-coded, we need to evaluate the system-wide utility for both cases, and take the maximum of them. Computing the system-wide utility for each of these smaller subproblems and coding granularity leads to the optimal solution for subproblem $P(2,2)$.

This example reveals that solving l layer subproblems requires optimal solutions for $l-1$ layer subproblems. Therefore, we sequentially solve subproblems with 1 layer, 2 layers, until L layers. Notice that, we assign rates and granularity in descending order. That is, we first determine the optimal rate and granularity for layer L by solving subproblem $P(c,1)$, where $1 \le c \le C$. We then recursively search for the optimal rates and granularity for layers $L-1, L-2, \ldots, 1$.

We present the details on solving a general subproblem $P(c,l)$, where $1 \le c \le C$ and $1 \le l \le L$ in the following. We assume the effective rate of each client class is used as its utility in our discussion. That is, we employ this utility function: $u_{\text{rate}}(\bar{b}_c, b_c) = \bar{b}_c$. We, however, emphasize that our algorithm works with any utility function as we will demonstrate in Section V. We first solve subproblem $P(c,1)$ for $1 \le c \le C$. The previous two 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, we compute the optimal system utility $Y^*(c,1)$ as follows:

$$Y^*(c,1) = \begin{cases} \sum_{k=c}^{C} b_c f(k), & L=1 \\ \sum_{k=c}^{C} \frac{b_c}{1+a_0(b_c)} f(k), & L \neq 1 \end{cases}. \qquad (5)$$

In the above equation, the first line represents the case where layer 1 is indeed the base layer that has zero scalability overhead. The second line represents the case where layer 1 is not a base layer, and thus its width is scaled down using the overhead function. In the latter case, we assume layer 1 is CGS-coded, however, we have not determined its granularity. The granularity of layer 1 will be addressed in a moment.

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. These subproblems are located at lower right part in Fig. 3 in our illustrative example. Next we explain how to construct an l layer structure from an $l-1$ layer structure.

We use $\hat{S} = \{(\hat{r}_i, \hat{g}_i), i = 1, 2, \ldots, l-1\}$ to denote an optimal stream structure for subproblem $P(i,l-1)$. We add a CGS-coded layer at rate b_c into this optimal solution to form an l layer structure. Since i is always equal to or larger than $c+1$, the rate b_c is lower than the rates of all layers in an optimal solution for $P(i,l-1)$. Therefore, we use r_1 to denote the rate of this new layer, and let $r_{k+1} = \hat{r}_k$, where $k = 1, \ldots, l-1$. That is, we write $S = \{(r_i, g_i), i = 1, 2, \ldots, l\}$ as an l-layer stream structure. We observe that this insertion of layer 1 classifies client classes into three categories. For clients in classes b_{c-1} and below, they do not receive any layer even after adding layer 1. That is, the system utility contributed by them remains zero. For clients in classes $b_c, b_{c+1}, \ldots, b_{i-1}$, they start receiving streams because of the addition of layer 1, and thus they contribute additional system utility. We define this additional system utility as function $U(c,l,i)$, which is given as follows:

$$U = \begin{cases} b_c \sum_{k=c}^{i-1} f(k), & l = L, g_2 = 0 \\ \sum_{k=c}^{i-1} \left[b_c + \frac{b_k - b_c}{1+a_1(b_i)} \right] f(k), & l = L, g_2 = 1 \\ \frac{b_c}{1+a_0(b_c)} \sum_{k=c}^{i-1} f(k), & l \neq L, g_2 = 0 \\ \sum_{k=c}^{i-1} \left[\frac{b_c}{1+a_0(b_c)} + \frac{b_k - b_c}{1+a_1(b_i)} \right] f(k), & l \neq L, g_2 = 1 \end{cases}. \qquad (6)$$

The above equation indicates that the additional system utility depends on the granularity of layer 2. The first two lines consider that layer 1 is the base layer that has no scalability overhead, while the other two lines assume layer 1 is CGS-coded. For clients in class b_i and above, they receive layer 2 before and after the addition of layer 1. Their effective rates, however, are reduced because of the addition of layer 1 as scalability overhead is a nonincreasing function. We define the effective rate reduction as function $D(c,l,i)$, which is given as follows:

$$D = \begin{cases} \frac{b_i}{1+a_0(b_i)} - b_c - \frac{b_i - b_c}{1+a_0(b_i)}, & l = L, g_2 = 0 \\ \frac{b_i}{1+a_0(b_i)} - b_c - \frac{b_i - b_c}{1+a_1(b_i)}, & l = L, g_2 = 1 \\ \frac{b_i}{1+a_0(b_i)} - \frac{b_c}{1+a_0(b_c)} - \frac{b_i - b_c}{1+a_0(b_i)}, & l \neq L, g_2 = 0 \\ \frac{b_i}{1+a_0(b_i)} - \frac{b_c}{1+a_0(b_c)} - \frac{b_i - b_c}{1+a_1(b_i)}, & l \neq L, g_2 = 1 \end{cases}. \qquad (7)$$

The above equation follows the definition of effective rates. The first term represents the effective rate for these clients before the addition of layer 1. The second and third terms account for the effective rate after layer 1 is added. Again, the effective rate reduction depends on the granularity of layer 2.

The analysis of these categories of client classes allows us to compute the system-wide utility after adding a CGS-coded layer to the optimal structure of subproblem $P(i,l-1)$. Therefore, we can use the following formulation to solve subproblem $P(c,l)$ by utilizing optimal solutions for smaller subproblems:

$$Y^*(c,l) = \max_{\substack{c+1 \leq i \leq C-l+1, \\ g_2 \in \{1,0\}}} \left\{ Y^*(i,l-1) - D(c,l,i) \sum_{k=i}^{C} f(k) + U(c,l,i) \right\} \qquad (8)$$

where $Y^*(c,l)$ represents the optimal system utility for subproblem $P(c,l)$.

The formulation in (8) updates the maximal system utility of $P(i,l-1)$ by deducting utility $D(c,l,i)$ from all clients in class i and above, and adding the additional utility function $U(c,l,i)$. It 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 (8). An important property of (8) 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 $P_0(C,L)$ based on optimal solutions for subproblems $P(c,l)$.

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 $P_0(C,L)$ is the one that achieves maximum system utility: $Y_0^* = \max_{1 \leq c \leq C-L+1}\{Y^*(c,L)\}$.

Proof: Let S^* be an optimal structure for problem $P_0(C,L)$, and r_1^* represent the coding rate of layer 1 in S^*. Clearly, we have $r_1^* \in [0, b_C]$. This is because r_1^* can not be larger than b_C, otherwise none of the client classes receives anything, and thus S^* can not be an optimal structure. We divide the range $[0, b_C]$ into nonoverlapping intervals $(b_{c-1}, b_c]$, where $c = 1, 2, \ldots, C$ and $b_0 = 0$. Now assume that the optimal rate r_1^* occurs in an arbitrary interval $(b_{z-1}, b_z]$, for some $1 \le z \le C$. Because the constraint $b_{z-1} < r_1^* \le b_z$ is satisfied, we know that the optimal structure for subproblem $P(z, L)$ leads to the maximum system wide utility for problem $P_0(C, L)$. As r_1^* must fall in exactly one of these intervals, we solve subproblems $P(c, L)$, for $c = 1, 2, \ldots, C$, then compute the Y_0^* by: $Y_0^* = \max_{1 \le c \le C}\{Y^*(c, L)\}$, which results in the maximal system utility Y_0^* for the original problem $P_0(C, L)$. Furthermore, Lemma 1 allows us to ignore subproblems $P(c, L)$, where $c = C - L + 2, C - L + 3, \ldots, C$. ■

The above theorem illustrates that the optimal structure for problem $P_0(C, L)$ can be derived by finding the maximal system-wide utility among all optimal solutions for subproblems $P(c, L)$, where $1 \le c \le C - L + 1$, while the optimal solutions for subproblems $P(c, L)$ can be found by iteratively solving smaller subproblems. Next, we present the details of our algorithm.

The pseudo-code of our algorithm is shown in Fig. 4. The algorithm takes the following inputs: client class distribution f_c, client network bandwidth b_c, and number of layers L. The algorithm also requires specifying the utility function and the scalability overhead function. It produces the optimal rate r_l^* and granularity g_l^* (CGS or FGS) for each layer l. The algorithm works by sequentially solving subproblems $P(c, l)$ for $1 \le l \le L$ and $1 \le c \le C$. It re-uses the optimal solutions of $(l-1)$-layer structuring subproblems to find the l-layer optimal solutions. We first solve the 1-layer structuring subproblem in lines 2–4. We then solve all other subproblems with more than one layer. We do this by using the two for loops starting at lines 6 and 7, which go through layers and client classes, respectively. We solve subproblem $P(c, l)$ as follows. The loop in lines 10–19 takes an optimal $l - 1$ layer structure produced by previous subproblems $P(i, l - 1)$, adds a layer to it, and computes the system-wide utility of this new l-layer structuring policy. To determine the optimal granularity, it computes a CGS-coded utility in line 11 and an FGS-coded utility in line 12. In lines 14–19, we check whether the current i leads to a structuring policy with higher system-wide utility than the best known l-layer structure. In line 21, we find the optimal structuring policy for problem $P_0(C, L)$ based on the solutions for subproblems using Theorem 1. Finally, lines 23–28 find the optimal L-layer stream structure using backtracking technique.

The following theorem gives the time and space complexities of our algorithm.

Theorem 2 (Complexity): The time complexity of the ScsOpt algorithm in Fig. 4 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.

Proof: The algorithm uses two for-loops to sequentially solve subproblem $P(c, l)$, which take $O(CL)$. For each $P(c, l)$, we check $C - l + 1$ previous subproblems, which takes another $O(C)$. For each previous subproblem, we compute its system-wide utility in $O(C)$ steps. Therefore, the time complexity of our algorithm is $O(C^3L)$. It is easy to see that the space complexity for our algorithm is $O(CL)$, because the data structures used are: $y(c, l)$ for storing optimal system utility; $r(c, l)$ for storing optimal rate; $g(c, l)$ for storing optimal granularity; and $n(r, l)$ for back-tracking. All these data structures are 2-dimensional arrays, where $1 \le c \le C$, and $1 \le l \le L$. ■

Algorithm ScsOpt

```
1.  /*Solve subproblem $P(c,1)$ for all $1 \le c \le C$*/
2.  for $c$ = 1 to $C$
3.    Compute $y(c,1)$ using Eq. (5);
4.     $g(c,1) = 0$; $r(c,1) = b_c$;
5.  /*Solve subproblem $P(c,l)$ for $2 \le l \le L$,
5.   $1 \le c \le C - l + 1$*/
6.  for $l$ = 2 to $L$
7.    for $c$ = $l$ to $C - l + 1$
8.      $y(c,l) = -\infty$;
9.      /*Compute $Y^*(c,l)$ using Eq. (8) and solutions
9.       of previous subproblems $P(i,l-1)$*/
10.     for $i = c + 1$ to $C - l + 1$
11.       Compute $util\_cgs = Y^*(i,l-1)$ using Eqs.
11.       (6-8) with $g_2 = 0$;
12.       Compute $util\_fgs = Y^*(i,l-1)$ using Eqs.
12.       (6-8) with $g_2 = 1$;
13.       /*Compare utility against known optimum*/
14.       if $util\_cgs > y(c,l)$
15.         $y(c,l) = util\_cgs$; $r(c,l) = b_c$;
16.         $g(c,l) = 0$; $n(c,l) = i$;
17.       if $util\_fgs > y(c,l)$
18.         $y(c,l) = util\_fgs$; $r(c,l) = b_c$;
19.         $g(c,l) = 1$; $n(c,l) = i$;
20. /*Find the optimal structuring for $P_0(C,L)$*/
21. $prev\_layer = \text{argmax}_{1 \le c \le C-L+1}\, y(c,L)$;
22. /*Find $r_l^*$ and $g_l^*$ for all $l$ using backtracking*/
23. $g_1^* = 0$;
24. for $l$ = $L$ to 1
25.   $r_{L-l+1}^* = r(prev\_layer, l)$;
26.   $g_{L-l+2}^* = g(prev\_layer, l)$;
27.   $prev\_layer = n(prev\_layer, l)$;
28. $r_L^* = r(prev\_layer, 1)$;
```

Fig. 4. Proposed algorithm to compute the optimal structure of scalable streams.

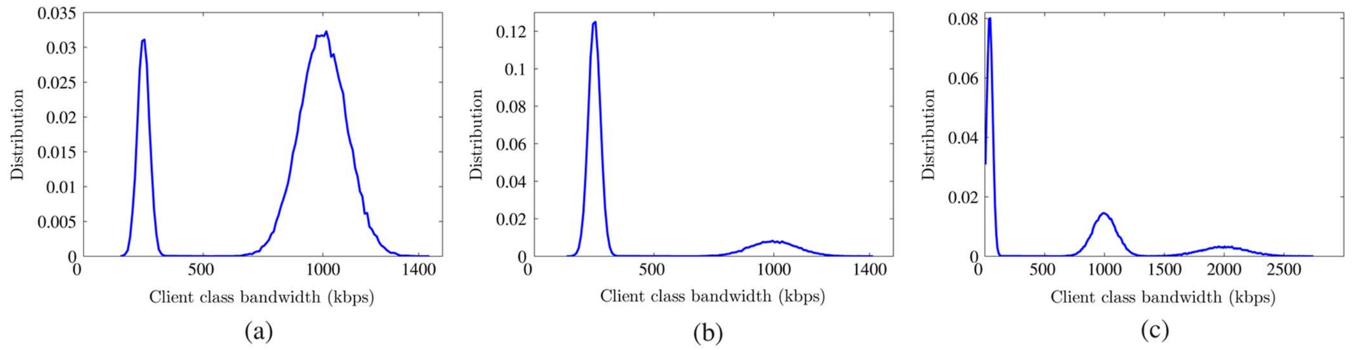

Fig. 5. Three of the client bandwidth distributions considered in the experimental study. The first scenario (uniform distribution) is not shown. (a) Scenario II: bi-modal (skewed to the right). (b) Scenario III: bi-modal (skewed to the left). (c) Scenario IV: Internet client distribution.

V. Evaluation

In this section, we first describe our experimental setup. Then we verify the optimality of our algorithm by comparing it against an exhaustive search algorithm. We then demonstrate that our algorithm allows various utility functions and produces optimal stream structures. That is followed by a study on the impact of choosing different utility functions. Next, we compare our algorithm with a widely used heuristic stream structuring algorithm. Last, we report the running time of our algorithm.

A. Setup

As mentioned in Section II, our algorithm would be run by a streaming server receiving requests from many concurrent clients for a specific scalable stream. The clients are heterogeneous in bandwidth. The server uses our algorithm and the client bandwidth information to determine the optimal structure of the stream. The output of our algorithm is the encoding rate and granularity (FGS or CGS) of each layer. This information is then fed to a video encoder such as H.264/SVC. We have implemented our algorithm in Java, and to rigorously evaluate its performance: i) we consider three different utility functions to quantify the optimal solution and ii) we simulate a large number of clients with different realistic bandwidth distributions. We elaborate on these parameters in the following.

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: i) $u_{\text{rate}}(\bar{b}_c, b_c) = \bar{b}_c$, which assumes that the higher the effective rate that a client receives, the more satisfied that client will be; ii) $u_{\text{utilization}}(\bar{b}_c, b_c) = \bar{b}_c/b_c$, 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 rate-distortion (R-D) curves to map the effective rate to perceived quality. For constructing R-D curves, we adopt a recent H.264/AVC R-D function which assumes that the transform coefficients are Cauchy distributed [23]. 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 [23] show that this model is more accurate than Laplacian and Gaussian based R-D models. We chose $u_{\text{psnr}}(\bar{b}_c, b_c) = -10\log_{10}[15.3787(0.1184\bar{b}_c)^{-2.2}]$ 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 consider 100000 clients with network bandwidth distributed according to four representative scenarios. Fig. 5 shows three of these distributions. The first scenario (not shown in the figure) is uniform between 35 and 3005 kbps. The second is a bi-modal 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-modal distribution is skewed to the right: 80% of client classes are from the normal distribution with mean 1000 kbps. Scenario III is a bi-modal 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. Scenario IV is a multimodal 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.

Finally, our algorithm can employ various 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 set $a_0(0) = 5\%$ 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 $a_1(r_l) = \max\{0.20 - 0.00004r_l, 0\}$.

B. Optimality of Our Algorithm

We compare the stream structures resulted by our algorithm against optimal solutions derived by an exhaustive search algorithm. We can only cover a few layers using exhaustive search due to the huge search space. We search for up to four layers optimal coding structures. We stop at four layer because the exhaustive search algorithm did not terminate in several hours for more layers. Fig. 6 shows the system-wide utility achieved by our algorithm and by the exhaustive search algorithm for all four

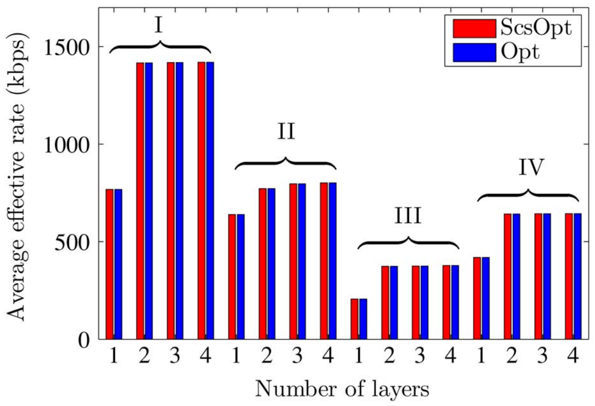

Fig. 6. Comparison between our algorithm (ScsOpt) and the optimal solution (Opt) derived by an exhaustive search algorithm for u_{rate} utility function. Results for scenarios I, II, III, and IV are shown from left to right. Similar results were obtained for the other two utility functions.

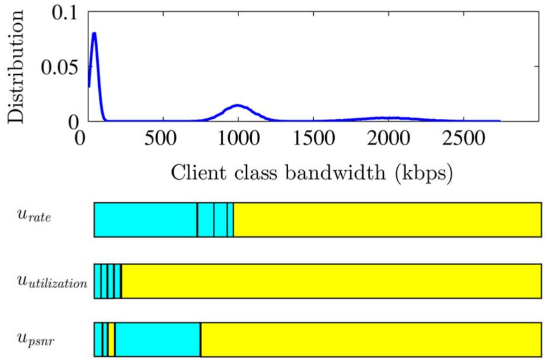

Fig. 7. Optimal structuring of a scalable video stream with 5 layers for scenario IV produced by our optimal algorithm with different utility functions.

scenarios and the u_{rate} utility function. This figure clearly confirms the optimality of our algorithm. Similar results were obtained for $u_{\text{utilization}}$ and u_{psnr} utility functions.

C. 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. 7 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 optimal 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) $u_{\text{utilization}}$: $\{(5, \text{CGS}), (15, \text{CGS}), (25, \text{CGS}), (35, \text{CGS}), (2745, \text{FGS})\}$, 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.

These results show that our algorithm is general and can be used with various utility functions in different environments. In the next subsection, we provide some guidelines on choosing the appropriate utility function.

D. Choosing Utility Functions

In this section, we shed some lights on the impact of using one utility function versus another. We do so by searching optimal stream structures for various scenarios and different number of layers. We show a sample result of constructing an optimal 4-layer stream structure for scenario IV as follows. In scenario IV, 50% of the clients have narrowband dial-up service, 35% have DSL service, and 15% have higher speed. Our algorithm produces the following stream structures for each utility function i) u_{rate}: $\{(735, \text{CGS}), (805, \text{CGS}), (855, \text{CGS}), (2745, \text{FGS})\}$, ii) $u_{\text{utilization}}$: $\{(5, \text{CGS}), (15, \text{CGS}), (25, \text{CGS}), (2745, \text{FGS})\}$, and iii) u_{psnr}: $\{(25, \text{CGS}), (35, \text{CGS}), (45, \text{CGS}) (2745, \text{FGS})\}$.

We plot the utility of individual classes in Fig. 8. The utility is derived by multiplying the client utility by the fraction of clients in that class. We first notice that the effective rate function favors broadband clients as indicated by Fig. 8(a). Actually the dial-up clients are completely ignored, as the first layer rate is set at 735 kbps. This is because a satisfied broadband client results in much higher effective rate compared to a satisfied narrowband client. In addition, increasing layer coding rates leads to lower scalability overhead thus higher effective rates. Consequently, narrowband clients are sacrificed. This poses a fairness issue among clients.

Using the bandwidth utilization function results in a fair stream structure as indicated by Fig. 8(b). This is because we use a relative utility function, in which the decision is not biased by the value of client bandwidth. Rather, the decision is made based on the mismatch between the client bandwidth and the receiving rate. We also observe a quite fair stream structure in Fig. 8(c). More resources are allocated to narrowband clients, compared to the effective rate case, because of the nonlinear R-D curve shape. The R-D curve increases dramatically at low rates, but saturates at high rates. Therefore, to maximize system-wide utility, more resources are allocated to clients that are receiving at lower rates.

These results help content providers to choose a utility function that suit their needs. For instance, a content provider would use the effective rate utility function if its customers are charged in terms of traffic amount. A nonprofit organization may be more interested in inter-client fairness and chooses bandwidth utilization function. Another content provider who wants to boost its client satisfaction would adopt the perceived quality function, which accommodates the nonlinear relationship between rate and actual perceived quality.

E. Comparison With Previous Structuring Algorithm

We compare the stream structures resulted by our algorithm against the heuristic structuring algorithm used in [15], [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.

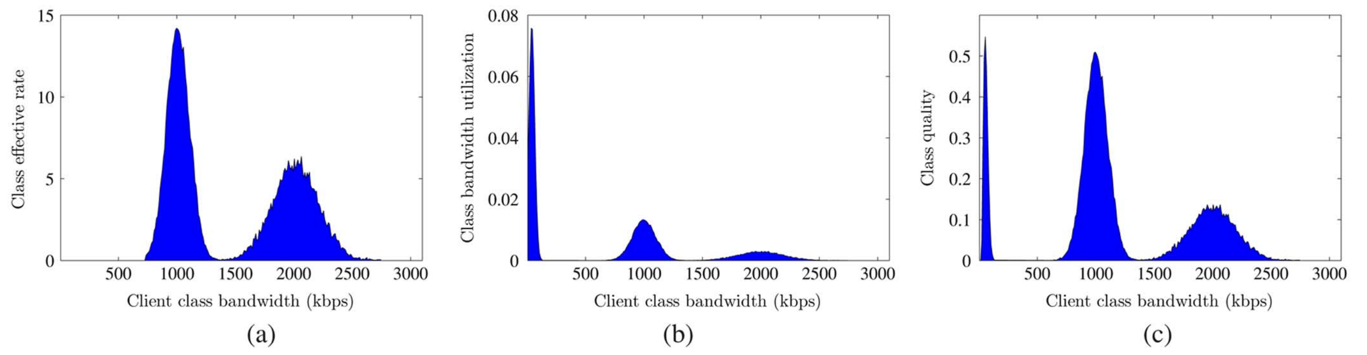

Fig. 8. Utility achieved by individual classes using our algorithm with various utility functions. (a) Effective rate. (b) Bandwith utilization. (c) Perceived quality.

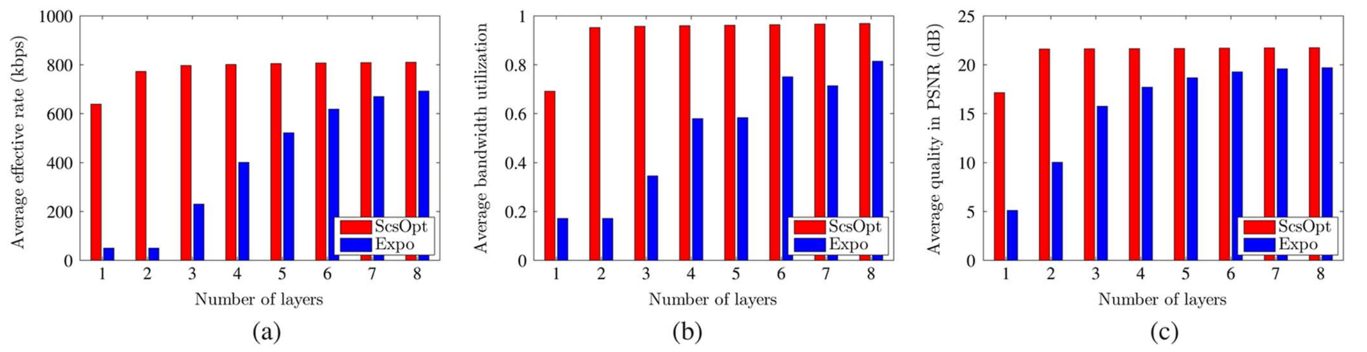

Fig. 9. Comparison between our algorithm (ScsOpt) and the heuristic algorithm (Expo) that exponentially allocates rates to layers. Sample results for different scenarios and various utility functions are shown. (a) Scenario II, using $u_{\rm rate}$. (b) Scenario III, using $u_{\rm utilization}$. (c) Scenario IV, using $u_{\rm psnr}$.

This covers a wide range of clients, from dial-up to broadband access links. We denote this algorithm by Expo in the plots. Fig. 9 illustrates the achieved system-wide utility by ScsOpt and Expo algorithms. This figure indicates that our algorithm outperforms the heuristic algorithm with significant margins. Similar results were obtained in all considered 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 Section IV-C and verified in Section V-C.

F. Running Time

While heuristic algorithms are typically efficient, they do not lead to optimal structuring policies. For example, the Expo algorithm produces significantly lower system-wide utility compared to our optimal algorithm. Therefore, we do not report the running time of heuristic algorithms.

Because of the huge search space, the exhaustive search algorithm consumes tremendous computational resources. For a moderate size problem with 100 client classes and 4 layers, the exhaustive search algorithm took at least 30 min to terminate. The running time is even longer if a complicated utility function—e.g., the perceived quality function—is employed. Consequently, the exhaustive search algorithm is not practical.

We present the running time of our algorithm in Table I. Our algorithm terminates in negligible time for straightforward utility functions: effective rate and bandwidth utilization. For instance, when using $u_{\rm rate}$ as utility function, our algorithm solves a problem with more than 200 client classes and eight layers in less than one second. In addition, our algorithm terminates in a few seconds even with a complicated perceived quality function. For example, our algorithm takes up to 2.5 seconds to solve a

TABLE I
RUNNING TIME (IN MILLISECONDS) OF OUR ALGORITHM WITH ALL CONSIDERED TEST SCENARIOS AND DIFFERENT UTILITY FUNCTIONS

Test	No. of	No. of layers L						
scenario	classes C	2	3	4	5	6	7	8
Using u_{rate} as utility function								
I	100	174	134	221	272	331	250	349
II	102	181	265	198	398	174	230	259
III	99	127	223	209	150	232	287	182
IV	222	230	326	397	506	755	857	741
Using $u_{utilization}$ as utility function								
I	100	150	136	168	209	204	395	236
II	102	141	260	293	259	214	314	225
III	99	146	166	208	445	198	188	362
IV	222	273	448	498	656	660	886	854
Using u_{psnr} as utility function								
I	100	309	650	994	1306	1631	1959	2426
II	102	449	867	1083	1430	1829	2104	2456
III	99	313	664	1017	1285	1579	1919	2220
IV	222	2547	6305	9722	13197	16768	20325	24935

problem with 100 client class and up to eight layers. Because of its low computational complexity, using our algorithm with more complex utility functions is feasible and practical. For instance, our algorithm can adopt an elaborate R-D function for higher estimation accuracy, and thus results in higher average perceived quality.

VI. 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 proposed an optimal algorithm to solve this problem. The 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, the proposed 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 experimentally verified that our algorithm produces the optimal results and runs in a few seconds on a commodity PC. 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.

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. For instance, a content provider can use the effective rate as a utility function if its customers are charged in terms of traffic amount. A nonprofit organization may be more interested in inter-client fairness and chooses a utility function based on the mismatch between client bandwidth and received rate.

References

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

[2] *Joint Scalable Video Model Reference Software*, JSVM 8.0, Feb. 2007, Joint Video Team.

[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 Trans. Multimedia*, vol. 3, no. 1, pp. 53–68, Mar. 2001.

[4] W. Li, "Overview of fine granularity scalability in MPEG-4 video standard," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 11, no. 3, pp. 301–317, Mar. 2001.

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

[6] J. Liu, B. Li, and Y. Zhang, "Adaptive video multicast over the Internet," *IEEE Multimedia Mag.*, vol. 10, no. 1, pp. 22–33, Jan. 2003.

[7] X. Li, M. Ammar, and S. Paul, "Video multicast over the Internet," *IEEE Network Mag.*, vol. 13, no. 2, pp. 46–60, Mar. 1999.

[8] S. Cheung, M. Ammar, and X. Li, "On the use of destination set grouping to improve fairness in multicast video distribution," in *Proc. IEEE INFOCOM'96*, San Francisco, CA, Mar. 1996, pp. 553–560.

[9] S. McCanne, V. Jacobson, and M. Vetterli, "Receiver-driven layered multicast," in *Proc. ACM SIGCOMM'96*, Palo Alto, CA, Aug. 1996, pp. 117–130.

[10] X. Li, S. Paul, and M. Ammar, "Multi-session rate control for layered video multicast," in *Proc. ACM/SPIE Multimedia Computing and Networking (MMCN'99)*, San Jose, CA, Jan. 1999, pp. 175–189.

[11] L. Vicisano, L. Rizzo, and J. Crowcroft, "TCP-like congestion control for layered multicast data transfer," in *Proc. IEEE INFOCOM'98*, San Francisco, CA, Mar. 1998, pp. 996–1003.

[12] T. Jiang, M. Ammar, and E. Zegura, "On the use of destination set grouping to improve inter-receiver fairness for multicast ABR sessions," in *Proc. IEEE INFOCOM'00*, Tel Aviv, Israel, Mar. 2000, pp. 42–51.

[13] N. Shacham, "Multipoint communication by hierarchically encoded data," in *Proc. IEEE INFOCOM'92: Conf. Comput. Commun.*, Florence, Italy, May 1992, pp. 2107–2114.

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

[15] 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. IEEE INFOCOM'03*, San Francisco, CA, Mar. 2003, pp. 630–640.

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

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

[18] 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. Int. Packet Video Workshop (PV'01)*, Kyongju, Korea, Apr. 2001.

[19] T. Kim and M. Ammar, "A comparison of layering and stream replication video multicast schemes," in *Proc. ACM Int. Workshop Network Oper. Syst. Support for Digital Audio Video (NOSSDAV'01)*, Port Jefferson, NY, Jun. 2001, pp. 63–72.

[20] T. Kim and M. Ammar, "A comparison of heterogeneous video multicast schemes: layered encoding or stream replication," *IEEE Trans. Multimedia*, vol. 7, no. 6, pp. 1123–1130, Dec. 2005.

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

[22] C. Hsu and M. Hefeeda, "Optimal partitioning of fine-grained scalable video streams," in *Proc. ACM Int. Workshop Network Oper. Syst. Support for Digital Audio Video (NOSSDAV'07)*, Urbana-Champaign, IL, Jun. 2007, pp. 63–68.

[23] 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 Trans. Circuits Syst. Video Technol.*, vol. 15, no. 8, pp. 994–1006, Aug. 2005.

Cheng-Hsin Hsu received the M.Eng. degree from the University of Maryland, College Park, in 2003, and the M.Sc. and B.Sc. degrees from National Chung-Cheng University, Taiwan in 2000 and 1996, respectively. He is currently pursuing the Ph.D. degree in the School of Computing Science, Simon Fraser University, Surrey, BC, Canada.

His research interests are in the area of multimedia networking and scalable video coding.

Mohamed Hefeeda (S'01–M'04) received the M.Sc. and B.Sc. degrees from Mansoura University, Egypt, in 1997 and 1994, respectively, and the Ph.D. degree from Purdue University, West Lafayette, IN, in 2004.

He is an Assistant Professor in the School of Computing Science, Simon Fraser University, Surrey, BC, Canada, where he leads the Network Systems Lab. His research is funded by Canadian funding agencies and industry through several grants. His research interests include multimedia networking, peer-to-peer systems, and wireless sensor networks.

Dr. Hefeeda is a member of the ACM Special Interest Groups on Data Communications (SIGCOMM) and Multimedia (SIGMM).