

Techniques for Efficient Streaming of Layered Video in Heterogeneous Client Environments

Aravindan Raghuv eer

Dept. of Computer Science & Engg.
University of Minnesota
Minneapolis, MN 55455
Email: aravind@cs.umn.edu

NamOh Kang

School of Computer Science & Engg.
Chung-Ang University
South Korea
Email: kang@archi.cse.cau.ac.kr

David H.C. Du

Dept. of Computer Science & Engg.
University of Minnesota
Minneapolis, MN 55455
Email: du@cs.umn.edu

Abstract—Universal Multimedia Access (UMA) refers to accessing multimedia content over a wide range of client terminals and network capacities. Scalable coding is a very popular technique to enable UMA for video. Overhead introduced by the scalable coding approach limits the number of layers that can be stored for each video. Therefore some clients may be served the closest available quality than the best-fit quality. This is a major drawback of scalable coding from the end-user perspective. We propose to employ transcoding to tailor content exactly to the client’s best-fit quality when the required layer is not stored. Inserting a transcoder in the server-client path introduces new challenges in deciding the layering structure (number of layers, bandwidth per layer) of a video. The optimal layering structure should be decided based on factors like total I/O bandwidth penalty incurred due to layering and transcoding effort required to service the “non-layered” versions. The solution to this problem is further complicated by practical issues like diverse popularity of video objects and resource availability. Another issue that we address in this paper is reducing WAN bandwidth penalty incurred due to transport and coding overhead inherent to scalable coding. This particular problem applies to all schemes that use layered encoding to broadcast video. We map the above mentioned problems onto a 0-1 multiple choice knapsack structure and propose an algorithm to find a near-optimal solution. The uniqueness of our approach not only lies in the streaming model but also in the integrated manner in which we address a variety of issues put forth by layered coding.

I. INTRODUCTION

With the advent of low power processors, display devices and widespread wireless access, the variety of multimedia consumers has been rapidly increasing. Layered encoding and transcoding are two popular approaches that are used to serve heterogeneous terminals. So far the scientific community has often viewed them as *competing* solutions. In a new perspective, we propose to use both layering and transcoding in *complementary* roles. While the server streams layered video, a LAN (local area network) agent performs transcoding (if needed) before reflecting the stream to the client. We observe that this model provides some powerful benefits that neither scalable coding nor transcoding alone can offer. We explain one such key advantage in the following paragraph.

It has been shown that layered coding incurs bandwidth overhead and only a limited number of layers should be stored to achieve the benefits of layering [1]–[3]. Therefore choosing the layering structure is an important issue. [1]

provides an algorithm to find the optimal layering structure of a video. However by choosing only a few layers, some receivers experience lower quality than what they can handle. We propose to overcome this problem by using transcoding to service those quality levels not covered by layering. Thus we use transcoding to *mask* the discrete quality levels and scalable coding to use server resources efficiently and to elegantly adapt to network conditions. By introducing a transcoder between the server and the client, new issues arise in the optimal layering structure problem. The layering structure of a video determines the amount of transcoding effort required. Since transcoding is CPU intensive and processing power is a finite resource, we design an algorithm that chooses layers to minimize the transcoding effort involved while adhering to resource constraints at the server.

The last problem we explore in this paper is to minimize the WAN bandwidth penalty due to layering overhead. Recent advances [4] have made layered coding more flexible. We propose a technique that exploits such flexibility to dynamically vary the sizes of layers (without affecting the layering structure on the disk) to reduce the WAN bandwidth overhead. Our simulations show that significant WAN bandwidth savings can be achieved with the proposed algorithm.

II. RELATED WORK

Layered coding and transcoding are two different approaches that address the same problem: “How to achieve Universal Multimedia Access?” In the scalable coding approach, a video is partitioned into a base layer and multiple enhancement layers. The base layer contains the most important coding information and each enhancement layer further refines the video quality [5]–[7]. Receivers choose the base layer and one or more enhancement layers based on their capability. On the other hand, transcoding is a technique to alter the bitrate of a video stream to exactly suit the receiver capability.

In layered coding, the client receives the base layer and one or more enhancement layers, combines and decodes them. We refer to the task of combining the base and enhancement layers into a single video stream, as *composing*. The main advantage of scalable coding is that it uses the storage resources at the server very efficiently by storing only the differential components across multiple quality versions of a video. Also

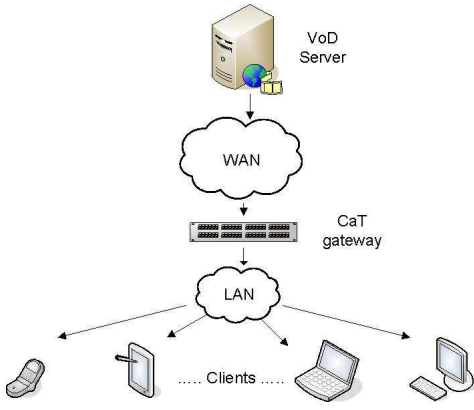


Fig. 1. The Cascade Streaming Architecture

adaptation to network conditions can be elegantly handled by adding or dropping enhancement layers [5]. However, the disadvantage of layered coding techniques in popular standards like MPEG-2 [6], is that quality levels are discrete and static. Therefore they cannot provide the best possible quality across a wide variety of terminals. Recent advances in video coding [4] allow dynamic manipulation of the layering structure with very low response times. The primary drawback of such more recent schemes is that they still incur significant bandwidth overhead [2], [3]. [1] shows that when layering overhead is considered, only a limited number of layers should be stored to achieve maximum perceived video quality. This is because with more layers, the benefits due to finer granularity is diminished by the layering overhead. [1] further provides efficient algorithms to find the optimal number of layers for videos to maximize quality for a given network bandwidth budget. However with limited number of layers for each video, there will be a mismatch between the required quality and the achieved quality for some receivers. In this paper we propose to remove this mismatch by inserting a transcoding element in the server-client path. Inserting a transcoder introduces new issues in finding the optimal layering structure. This is another aspect where our work differs from [1].

Recent architectures [8] can perform transcoding without *decompressing* the video and hence vastly reducing the complexity of transcoders. This has enabled real time or on-the-fly transcoding [9], [10]. Transcoding offers multiple advantages over scalable coding. It can help “tailor” content to fit client’s requirement in any dimension (frame rate, resolution etc). Transcoding is a vital component to provide access to MPEG-2 content (representing a vast majority of existing content) to diverse multimedia devices and hence is predicted to be an active research area for many more years [8]. A disadvantage of the transcoding approach is that the computation effort required is significantly larger than in scalable coding.

III. SYSTEM MODEL

In the previous section we observed that the layered encoding approach is best suited for the server and WAN environments because of its scalability and adaptability properties

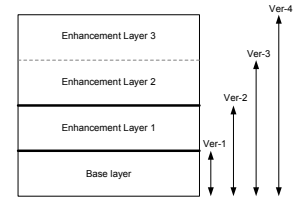


Fig. 2. Selective Layering: Thick lines denote stored layers, dotted line denotes unstored layer

respectively. On the other hand, the transcoding approach is best suited to tailor content to fit the client exactly. In this section, we explore a unified model that uses both layering and transcoding and thereby exploiting their strengths. We refer to this unified model as the Cascade Streaming Architecture (CSA). In CSA, the video server streams layered video and an intermediate LAN agent called the Composing and Transcoding (CaT) gateway [11] performs transcoding of the layered stream before reflecting it to the client (refer Figure-1). The underlying philosophy is to employ an approach in an environment (or role) where it is best suited and most efficient.

We now illustrate the functioning of CSA with two scenarios. Assume that we need to service a video to 4 types of clients. So there are 4 possible layers. But due to resource constraints at the server, we store only 3 layers as shown in Figure (Enhancement layers 2,3 are stored merged as enhancement layer 2&3)

1) *Scenario 1:* A client sends a request for version 3 to its CaT gateway, which then forwards this request to the VoD server. Since the server has not stored the enhancement layer for version 3, it sends the layers corresponding to the closest higher quality version available, viz. version 4 (layers 0,1,2&3), to the CaT gateway. The gateway then composes the layers, transcodes version 4 to version 3 and forwards it to the client.

2) *Scenario 2:* A client requests version 2 and since the server has exact layers corresponding to version 2, it sends them (layers 0,1) to the CaT gateway. In this case, the gateway does not need to perform transcoding. It composes the layers into a single video object and forwards it to the client.

CSA makes transcoding and layering truly complementary, with each method alleviating disadvantages of the other. Transcoding is made less CPU intensive because of the availability of closer higher quality versions provided by layering while discrete quality levels of layered coding are *masked* by transcoding.

IV. CHALLENGES IN CASCADE STREAMING

In this section, we first analyse the effect of layered encoding on the VoD server resources. We then define the issues that need to be addressed to manage these limited resources efficiently.

Layering overhead consumes three resources at the server: storage, I/O bandwidth and WAN bandwidth. In this study, we assume that storage is less of a bottleneck when compared

to I/O and WAN bandwidths. Layered coding affects I/O and WAN bandwidth consumption as follows:

a) *I/O Bandwidth*: [3] reports that the I/O bandwidth requirement of layered video can be 20% - 30% higher than unlayered video. Also the layering structure of a video affects the percentage of bandwidth wasted on overhead [1].

b) *WAN bandwidth*: When video is serviced as layers, a significant fraction of the WAN bandwidth is spent on packetization and protocol overhead [2].

At the server, both I/O and WAN bandwidths are limited resources and hence need to be managed efficiently. In the next two sections we address the following questions in the context of the *Cascade Streaming Architecture*:

- 1) How to decide the optimal layering structure of the videos such that the server's I/O bandwidth is used efficiently? (Selective Layering: Section V)
- 2) How to minimize the WAN bandwidth wastage due to layered coding? (On-the-fly Composing: Section VI)

Notations: We now define the notations that we will follow throughout this paper. A video i of version j is termed as a video object and is denoted by V_{ij} . Each video has m_i versions. s_{ij} is the size (in bytes) and d_{ij} represents the duration (in seconds) of V_{ij} . Popularity α_{ij} is defined as the number of concurrent accesses per unit time to object V_{ij} .

V. SELECTIVE LAYERING

Very fine quality variations can be provided by having a large number of layers. But this advantage is overshadowed by the fact that more layers will incur more overhead. The negative effect of overhead is accentuated by the effect of popularity. If a high quality version of a video (a base layer with multiple enhancement layers) is very popular, then the I/O bandwidth spent on layering overhead can become prohibitively high. Therefore we need to limit the number of layers to store for each video. This problem is further complicated by the structure of CSA. As shown in Scenario-1 of Section III, the CaT gateway might need to perform transcoding if the server does not store an exact match to a requested version. Therefore the position of the layers (bandwidth of layers) is crucial in deciding the amount of transcoding effort spent at the CaT gateway. Since CPU is a critical resource at the gateway, the layers should be chosen to minimize the transcoding effort required. Also we observe that the I/O bandwidth requirement of a video initially decreases as more layers are added. But beyond a cutoff point, the I/O requirement increases with the number of layers. This can be attributed to the coding overhead playing a dominant role in determining the I/O requirement. For a more detailed analysis of the tradeoffs in the CSA model we refer the reader to [12].

The *selective layering* problem deals with deciding the optimal layering structure for each video to minimize the transcoding effort while using the server's I/O bandwidth efficiently. In section V-A, we first introduce some important definitions and notations. In section V-B, we then discuss the problem of *optimal k-set layering* and propose an algorithm to solve it. We then use *optimal k-set layering* as the foundation

to solve the *selective layering* problem in Sections V-C and V-D.

A. Definitions

- 1) **Layering Set (L_i)**: The set of points at which video i is partitioned. If video i has m_i versions, then there are a total of $(m_i - 1)$ such possible layering points.
- 2) **Size of Layering Set (SL_i)**: The number of layering points for video i . It can vary between 0 (no layers) to $m_i - 1$ (all versions have their corresponding layers).
- 3) **Transcoding Cost ($t_{ix \rightarrow y}$)**: The CPU effort required to transcode a video object i from version x to version y . We model the transcoding cost as follows:

$$t_{ix \rightarrow y} = K_T \cdot \alpha_{iy} \cdot f(x, y) : s_{ix} > s_{iy} \quad (1)$$

where K_T is a constant of proportionality and f represents the CPU effort per transcoding operation. In our experiments in Section V-E, we model $f(x, y) = (x - y)$.

- 4) **Transcoding Cost Savings (T_{ij})**: The CPU effort that would be saved by having a layering set of size j for video i compared to having a layering set of size 0.

$$T_{ij} = \sum_{(a \notin L_i)} (t_{im_i \rightarrow a} - t_{il_a \rightarrow a}); l_a \in L_i \quad (2)$$

($a \notin L_i$) represents versions that need to be transcoded (ex. version 3 in Scenario-1 of Section III) and l_a is the next higher quality version to transcode from.

B. The optimal k-set layering problem

The optimal k-set layering problem can be stated as follows: "For a given video i find the layering set of size k that provides the maximum possible transcoding costs savings."

This problem is not guaranteed to have the optimal substructure property always. Consider a video that has 4 versions (3 layering points). Assume that all the layers have the same size and all versions have the same probabilities of access = 0.25. If $k = 1$, then the optimal layering set is $\{2\}$. If $k = 2$, then the optimal layering set is $\{1, 3\}$. So the solution set to $k = 2$ does not contain the solution set to a subproblem ie. $k = 1$.

Since optimal substructure does not hold always, techniques like dynamic programming and greedy approach cannot be applied to find the optimal solution. It can, however, be found by explicitly enumerating all possible combinations. This brute force approach has a worst case running time of $O(2^n)$ to find the optimal k-sets for all k , $1 \leq k \leq n$. We therefore design a simple greedy algorithm, of time complexity $O(n^2)$, by relaxing the optimal substructure constraint.

The greedy algorithm can be described as follows: In every step we add that layering point, to the current set $\{R\}$, which increases the transcoding benefit of $\{R\}$ by the maximum amount. We repeat this until k choices are made and report the overall transcoding cost savings, T_{ik} for the set $\{R\}$.

In [12] we study the factors that affect the optimality of the greedy algorithm. It is seen that with skewed version access probabilities, there is more overlap in the problem substructure. This is because, layering points with the high

access probability should always be present in the solution set. As the “skewness” (controlled by a Zipf distribution) is increased, the accuracy of the greedy algorithm also increases.

C. Problem Formulation

The aim of selective layering is to find the *layering set* for each video i such that the overall transcoding cost savings is maximized and the I/O bandwidth constraint at the server (I_s) is not violated. This can be mathematically stated as follows:

$$\text{Maximize } \sum_{i=1}^n \sum_{j=0}^{m_i-1} T_{ij} \cdot X_{ij} \quad (3)$$

subject to:

$$\sum_{i=1}^n \sum_{j=0}^{m_i-1} I_{ij} \cdot X_{ij} \leq I_s \quad (C3.1)$$

$$\sum_{j=0}^{m_i-1} X_{ij} = 1, \text{ for } i = 1, 2, \dots, n \quad (C3.2)$$

$$X_{ij} \in \{0, 1\} \quad (C3.3)$$

where i is the video number ($1 \leq i \leq n$) and j is the size of the layering set for video i ($0 \leq j < m_i$). I_{ij} is the I/O bandwidth requirement for video i with layering set size of j . Let O_{im} be the overhead for the m^{th} layering point of video i .

$$I_{ij} = \sum_{k \in L_i} \frac{\alpha_{ik} \cdot S_{ik}}{d_{ik}} + \sum_{k \notin L_i} \frac{\alpha_{ik} \cdot S_{ij}}{d_{ik}} \quad (4)$$

$$S_{ik} = (s_{ik} + \sum_{(m \in L_i) \text{ and } (m < k)} O_{im})$$

Constraints C3.2 and C3.3 indicate that each video can have only one value for the size of the layering set from the set $\{0, 1, 2, \dots, m_i - 1\}$. A value of 0 implies no layering for that video and $m_i - 1$ indicates all layering points be chosen.

D. Solution to the Selective Layering Problem

The goal of the selective layering problem is to choose optimal layering set sizes for each video i (from the set $\{0, 1, 2, \dots, m_i - 1\}$) to maximize the transcoding cost savings subject to the server’s I/O bandwidth constraint. This can be mapped on to a 0-1 multiple choice knapsack problem (MCKP). In MCKP, multiple classes of items exist and the goal is to choose one item from each class to maximize the value of the knapsack while adhering to some resource constraint.

[13] presents a dynamic programming approach to find the exact solution to the 0-1 multiple choice knapsack problem. We use this approach to find the optimal layer set sizes for each video. Let $Z(i, y)$ denote the maximum transcoding cost savings with i videos and I/O bandwidth of y . A recursive equation in terms of Z can be written as follows:

$$Z(i, y) = \max_{0 \leq j < m_i} \{ \{T_{ij} + Z(i-1, y - I_{ij}) | y \geq I_{ij}\}, Z(i-1, y) \} \quad (5)$$

We start the recursion with the initialization $Z(0, y) = 0; \forall y \geq 0$ and proceed to obtain the value (and the corresponding layering sets) of $Z(n, I_s)$. In every step, we use optimal k -set layering to find the value of T_{ij} .

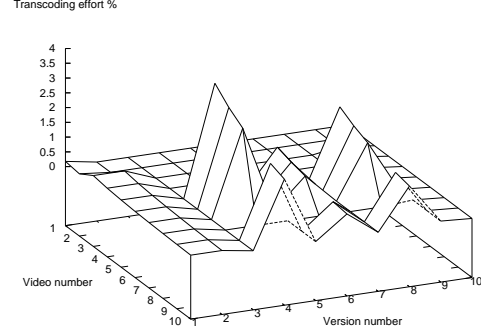


Fig. 3. Transcoding effort distribution

E. Experiments and Results

In this section, we conduct experiments to illustrate the behavior of the selective layering scheme. We also compare its performance with other popular layering schemes.

In our simulations, we consider 10 identical videos with the highest quality having an average bitrate of 0.734Mbps (Jurassic Park trace from [14]). We follow [1] to model the receiver capacities and layering overhead. We choose 10 receiver bandwidth capacities, drawn from a Gaussian distribution, that lie in the range 57.6 Kbps and 0.75 Mbps. This choice is justified because receiver capacities tend to be clustered rather than uniformly distributed. Since the layering overhead can vary based on the packetization scheme used, we study performance for layering overheads (per layer) in the range [0, 28.8] Kbps. We denote h as the fraction of maximum possible overhead. ie overhead = $h * 28.8 : 0 \leq h \leq 1$.

We model the popularity α_{ij} , at two levels for a more realistic treatment. At the first level, we draw the popularity of a video i , N_i from a Zipf distribution ($N_i = N_{tot} \cdot \frac{(1/i)^{(1-\theta)}}{\sum_{i=1}^n (1/i)^{(1-\theta)}}; N_{tot} = 100$) with skew parameter $\theta=0.271$. At the next level, we attach, to each receiver cluster j , an *access probability*, p_j . The popularity of a video object V_{ij} is then defined as : $\alpha_{ij} = N_i \cdot p_j$. We use the following access probabilities in our experiments: [0.02, 0.025, 0.04, 0.08, 0.4, 0.3, 0.075, 0.03, 0.02, 0.01]. The intuition behind the values chosen is that the access probability of a medium, medium-high quality receiver (like a TV/laptop) is, in practice, significantly larger than high end systems (like HDTV) or low capacity receivers like PDAs. In a practical setting, the *selective layering* algorithm is run periodically (every Δt time units). The popularity information collected during the previous period is used to estimate the popularity in the current period. N_i is the number of accesses to video i in a time duration Δt and p_j represents the fraction of accesses that were from a client cluster j during the same time window. Exponential smoothing is used to take history into account while predicting the video popularity and the receiver access probability for the current time window.

In the first experiment, we study the behavior of the selective layering algorithm. For convenience, videos are arranged in

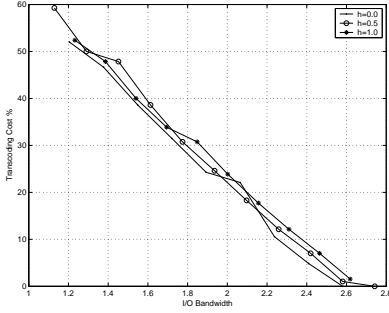


Fig. 4. Effect of layering overhead, I/O bandwidth

decreasing order of popularities ie. video-1 is the most popular and video-10, the least. Figure 3 shows the transcoding effort, required at the CaT gateway, for all video objects. It is seen that for highly popular videos(1, 2, 3) almost all layers are chosen to minimize the transcoding cost. Version 5 which has the highest access probability always has a corresponding layer for the same reason. These results clearly show that the selective layering algorithm prioritizes various layers of videos based on the transcoding savings that they can provide.

Next, we study the effect of overhead on the selective layering algorithm. Figure 4 shows the transcoding cost expended at the CaT gateway, for various values of overhead, available I/O bandwidth. We start with an initial I/O bandwidth I_{init} (shown as 1 on the x-axis) and keep increasing it. We see that the transcoding effort drops as more I/O bandwidth is available since more layers are added to each video. Larger values of layering overhead, h , need more I/O bandwidth to achieve the same amount of transcoding effort as with lesser overhead.

In the next experiment, we compare the greedy and optimal solutions to k -set layering with two popular layering schemes : additive layering and exponential/multiplicative layering [2], [5]. In additive layering, a video of bandwidth R is split into l layers by allocating a bandwidth of R/l to each layer. In exponential layering, the cumulative layer rates of adjacent layers are exponentially separated by a factor $\alpha > 1$. ie. $r_{i+1} = \alpha.r_i$. α is found as follows: $\alpha = \sqrt[l-1]{\frac{R}{r_b}}$ where r_b is the base layer bandwidth. Figure 5 shows the transcoding cost savings obtained with different number of layers in each scheme. Additive and exponential layering do not perform as well as the greedy heuristic because they do not take into account the receiver bandwidth distribution. On the other hand, the greedy heuristic plans the bandwidth allocated to each layer based on receiver distribution and popularities. We also see that the performance of the greedy algorithm closely follows that of the optimal algorithm (explicit enumeration).

VI. ON-THE-FLY COMPOSING

When a video is serviced as a base layer and multiple enhancement layers, some network bandwidth is wasted due to coding and packetization overhead [2], [4]. In this section, we present a technique to reduce this overhead. The intuition behind our approach is that it is possible to expand the base

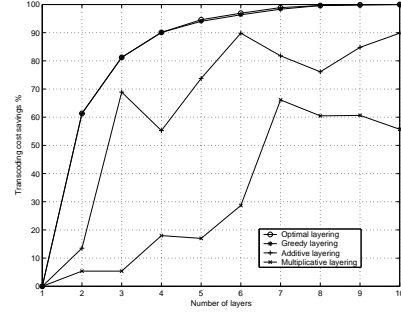


Fig. 5. Comparison of layering strategies

layer, on-the-fly, by stuffing one or more enhancement layers into it. The amount of overhead that would be eliminated is proportional to the number of layers composed into the original base layer. This does not affect the layering structure of the video on the disk but instead uses the server's CPU to dynamically increase the size of base layer and thereby reducing the number of enhancement layers (without affecting the video quality). With lesser number of enhancement layers, lesser network bandwidth overhead is incurred. Recent coding techniques [4] can perform such compaction with fast response times.

A. Problem Formulation

Each video object V_{ij} can fetch different bandwidth savings based on two factors : (i) popularity α_{ij} (ii) total number of layers composed (overhead avoided). For any video, the bandwidth saved increases with the number of layers composed and so does the CPU effort spent in composing. The challenge is to decide how to efficiently allocate the available CPU among all the video objects to maximize the bandwidth savings. Selective layering makes the problem harder because some video objects could be served using higher quality versions. We first address the issues that arise due to selective layering and then tackle the CPU allocation problem.

1) *Effect of Selective Layering*: Due to selective layering, a set of video objects (say $\{V\}$) may be serviced by sending out a higher quality video object (say V_{ik}). As far as the CPU resource allocation problem is concerned, there is no difference between servicing a request to video object V_{ik} and any object in the set $\{V\}$. We therefore adjust the popularity of V_{ik} as:

$$\alpha'_{ik} = \alpha_{ik} + \sum_{j \in \{V\}} \alpha_{ij}. \quad (6)$$

This popularity adjustment for video V_i is done for every version corresponding to each layering point in the layering set L_i and for the highest quality version m_i .

2) *Bandwidth Savings*: If every request for video object V_{ij} , is satisfied by composing the first k enhancement layers, then the amount of bandwidth savings (B_{ijk}) depends on the total overhead eliminated and the popularity α'_{ij} .

$$B_{ijk} = \frac{\alpha'_{ij}}{d_{ij}} \cdot \sum_{m=1}^{k-1} O_{im} \quad (7)$$

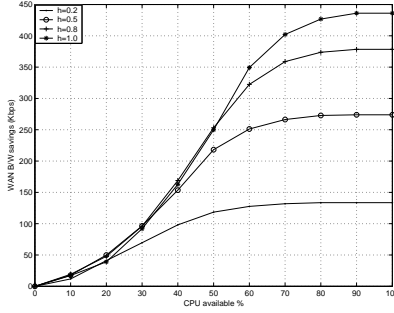


Fig. 6. Network Bandwidth savings using on-the-fly composing

3) *Composing Cost*: The CPU effort required to compose the first k layers of video object V_{ij} . We model the composing cost as follows (K_C is a constant of proportionality):

$$C_{ijk} = K_C \cdot \frac{\alpha'_{ij}}{d_{ij}} (s_{ik} + \sum_{m=1}^{k-1} O_{im}) \quad (8)$$

4) *Mathematical Formulation*: After video i is partitioned as explained in Section V-D, it has $SL_i + 1$ layers. For the sake of clarity, we will refer to these layers as 1, 2 ... $SL_i + 1$. The j^{th} layer is requested when a version that lies between $(j-1)$ and j^{th} layer is requested. This request can be satisfied by composing first k layers where $2 \leq k \leq j$. A compose operation is possible only when $j \geq 2$ because for a request to the first layer no composing is possible. Hence for the j^{th} layer, there are j composition possibilities. The CPU allocation problem deals with choosing the optimal k value for all layers in all videos. Optimality is defined in terms of overall WAN bandwidth savings that can be achieved for a given CPU availability C_s . This can be stated mathematically as follows:

$$\text{Maximize} \sum_{i=1}^n \sum_{j=1}^{(SL_i+1)} \sum_{k=1}^{k \leq j} B_{ijk} \cdot X_{ijk} \quad (9)$$

such that :

$$\begin{aligned} \sum_{i=1}^n \sum_{j \in S_i} \sum_{k \in S_i}^{k < j} C_{ijk} \cdot X_{ijk} &\leq C_s \\ \sum_{k \in S_i}^{k < j} X_{ijk} &= 1 \text{ for } i = 1, 2, \dots, n \text{ and } j \in S_i \\ X_{ijk} &\in \{0, 1\} \end{aligned}$$

Where i stands for the video number and j stands for the layer number within each video. Note that video i can have $(SL_i + 1)$ possible layers. k is the parameter that is used to test every j^{th} composition possibility for optimality. X_{ijk} (to be found by solving Equation 9) is 1 only when k is the optimal number of layers to be composed for video i , layer j .

B. Solution to the On-the-fly Composing Problem

This problem can be mapped on to a 0-1 multiple choice knapsack as done for the selective layering problem. Here instead of selecting one optimal element for each video (as in the selective layering case), we need to choose one optimal element for each layer in all videos. The solution is exactly same as the one described in Section V-D with minor differences in constructing the recurrence relation that we explain

below. The CPU allocation problem has 2 dimensions of search space: videos and layers while the selective layering had just one: videos. Therefore to construct the recurrence relation, we rearrange the problem from 2 dimensions (videos, layers) into a linear problem. There are totally $L_{tot} = \sum_{i=1}^n (SL_i + 1)$ contending layers in all. Let $Z(x, y)$ represent the optimal solution for x layers with y units of CPU available. The goal is to ultimately find $Z(L_{tot}, C_s)$. By making this change, the problem becomes exactly same in structure, as the selective layering problem and hence we can construct a recurrence relation to solve it as described before.

C. Experiments

We run composing optimization for the first experiment in Section V-E. Figure 6 shows the bandwidth saved by dynamic composing as a function of CPU availability for various layering overheads. The curve reaches a steady value when all layers are serviced by composing and no more bandwidth savings can be obtained. With the increase in layering overhead, the bandwidth savings also increases. In practice, the *on-the-fly composing* algorithm is run after *selective layering* is executed and the layers decided.

VII. CONCLUSION

In this study, we explored a new architecture that could build on the strengths of both layered coding and transcoding. We designed schemes that would enable efficient transport of layered video in three ways: a) Use transcoding to complement layered coding. b) decide an optimal layering structure for videos to make the architecture more scalable and manage I/O overhead. c) reduce layering overhead.

REFERENCES

- [1] J. Liu, B. Li, Y. T. Hou, and I. Chlamtac, "Dynamic layering and bandwidth allocation for multi-session video broadcasting with general utility functions," in *INFOCOM*, 2003.
- [2] M. Johanson and A. Lie, "Layered encoding and transmission of video in heterogenous environments," in *ACM Multimedia*, 2002.
- [3] T. Kim and M. H. Ammar, "A comparison of layering and stream replication video multicast schemes," in *ACM NOSSDAV*, June 2001.
- [4] W. Li, "Overview of fine granularity scalability in MPEG-4 video standard," *IEEE Trans. Circuits Syst. Video Techn.*, vol. 11, no. 3, 2001.
- [5] S. McCanne, V. Jacobson, and M. Vetterli, "Receiver-driven layered multicast," in *Proceedings ACM SIGCOMM*, vol. 26(4), 1996.
- [6] ISO/IEC, "Generic coding of moving pictures and associated audio information," in *ISO/IEC 13818-2*, 1995.
- [7] G. Feideropoulou, B. Pesquet-Popescu, and J. Belfiore, "Joint source-channel coding of scalable video," in *IEEE GLOBECOM*, 2004.
- [8] A. Vetro, C. Christopoulos, and H. Sun, "Video transcoding arch. and techniques: An overview," in *IEEE Signal Proc. Magazine*, March 2003.
- [9] Z. Lei and N. D. Georganas, "Video transcoding gateway for wireless video access," in *IEEE CCECE*, May 2003.
- [10] F. D. Angelis, I. Habiband, F. Davide, and M. Naghshineh, "Improving the delivery of multimedia services using an adaptive filtering strategy," in *IEEE GLOBECOM*, 2003.
- [11] E. Amir, S. McCanne, and H. Zhang, "An application level video gateway," in *ACM Multimedia*, 1995.
- [12] A. Raghuvver, N. Kang, and D. H. Du, "An architecture for universal multimedia access," University of Minnesota, Tech. Rep., 2005.
- [13] K. Dudzinski and S. Walukiewicz, "Exact methods for the knapsack problem and its generalizations," in *Euro. J. of Operations Res.*, 1987.
- [14] P. Seeling, M. Reisslein, and B. Kulapala, "Network performance evaluation using frame size and quality traces of single-layer and two-layer video: A tutorial," in *IEEE Communications Surveys and Tutorials*, 2004.