Reference Frame Optimization for Multi-path Video Streaming using Complexity Scaling

Gene Cheung
Hewlett-Packard Laboratories, Japan
Takaido Office Bldg.#3
3-8-13, Takaido-Higashi, Suginami-ku
Tokyo, 168-0072 Japan
Email: gene-cs.cheung@hp.com

Wai-tian Tan
Hewlett-Packard Laboratories, Palo Alto
1501 Page Mill Rd.
Palo Alto, CA
Email: wai-tian.tan@hp.com

***Abstract*— Recent video coding standards such as H.264 offer the flexibility to select reference frames during motion estimation for predicted frames. The application of traditional Lagrangian approach to the reference frame selection problem suffers from either bounded worst-case error but high complexity, or low complexity but undetermined worst-case error. In this paper, we propose a dynamic programming based optimization algorithm that admits trade-offs between computation complexity and worst-case error using advanced rounding techniques. We also evaluate the performance of the algorithm in the context of multi-path streaming over QoS-enabled networks. Results show significant streaming quality improvement over a static reference frame selection scheme.**

I. Introduction

The subject of this paper is to exploit the *reference frame selection* (RFS) feature of modern video coding standards to improve streaming quality over lossy networks. New video coding standards such as H.264 [1] offer many coding flexibilities for better coding and streaming performance. One of these flexibilities is flexible motion estimation support or RFS, where each predicted frame can choose among a number of frames for motion estimation. Often at the cost of lower coding efficiency, the ability to choose among multiple frames in the past for motion estimation can potentially avoid error propagation due to packet loss.

Given the available RFS feature of video coding, our goal is to tailor the reference frame selection for a given network streaming scenario. In this paper, we consider the scenario of multi-path streaming over QoS networks. By QoS networks, we mean networks which provide network layer QoS support like [2], where IP packets experience different packet loss rates at different service classes and different costs per packet. Here, we assume a cost constraint per transmission path. For networks without network-level QoS such as the Internet, end hosts can mimic a QoS network by applying forward error correction (FEC) of different strengths to different groups of packets. We consider a transmission rate constraint per transmission path in this case. We consider both cases under the same QoS network formulation in the paper.

By multi-paths, we mean two (or more) delivery paths are simultaneously available to end hosts for packet delivery during a streaming session. For wired networks, multi-paths can be made available, for example, as a result of application induced overlay routing or network supported source-based routing. For wireless networks, mobile terminals can conceivably be connected to two nearest base-stations via multiple antennas, or simultaneously using two network interfaces into two orthogonal wireless technologies such as wireless LAN and 3G wireless networks. One obvious advantage of multi-paths is the potentially larger combined transmission rate in the case when each path is rate constrained[1]. Another advantage can be better fault tolerance, where given the paths are disjoint and possess different and uncorrelated loss characteristics, simultaneous failure of both paths is less likely than a single path scenario.

We assume the application requires very low network delivery delay, to the extent that it cannot tolerate even one end-to-end packet retransmission. One reason can be that a small playback buffer is employed at the client side, together with the relatively large transmission delay of wireless links such as 3G cellular links [3], means that any retransmitted packet upon client request will miss its playback deadline and hence be rendered useless.

Given the described RFS feature of video coding and the network streaming scenario under consideration, the central problem is the following: for each predicted frame in a to-be-encoded video sequence, how to: i) select an appropriate reference frame for motion estimation, and ii) a QoS level and transmission path for packet delivery, such that the overall streaming performance is optimized? To this end, standard Lagrangian based optimization procedures can be employed. Nevertheless, there is no general mechanisms to simultaneously bound the running time of a Lagrangian optimization *and* bound the worst-case approximation error. As a result, there are practical challenges in using such optimization schemes for low-latency, quality-guaranteed media delivery. This paper offers an alternative optimization strategy, based on advanced integer rounding techniques, that provides a complexity and worst-case error bounded algorithm. Moreover, the algorithm is complexity scalable, where the quality of

[1]In some cases, using two transmission paths simultaneously decreases overall performance because of mutual signal interference. We assume here that the paths are orthogonal and therefore additive.

the obtained approximate solution can be traded off with the algorithm's running time.

The rest of the paper is organized as follows. After discussing related work in section II, we formulate the optimization problem and provide a solution in Section III and IV, respectively. A set of integer rounding-based procedures to reduce the complexity of the algorithm at the cost of solution quality are discussed in Section V. Results and conclusion are presented in Section VI and VII, respectively.

II. Previous Work

H.264 [1] is a new video coding standard that has demonstratably superior coding performance over existing standards such as MPEG-4 and H.263 over a range of bit rates. As part of the new standard definition is the flexibility of using any arbitrary frame to perform motion-estimation, originally introduced as Annex N in H.263+ and later as Annex U in H.263++. Early work on optimizing streaming quality using reference frame selection includes [4] [5]. Our work differs from previous works by employing a complexity-scalable optimization procedure and also applying optimization to jointly perform reference frame (RF) and transmission path (TP) selection.

A related research topic is multiple description (MD), where video is encoded into two (or more) "descriptions", and each description can be decoded independently of the other. For example, an MD stream can be obtained by coding the even frames into stream 1, and coding the odd frames independently from the even frames as stream 0. In [6], it is observed that when different descriptions are transmitted using different network paths, it is possible to apply error-concealment techniques at the decoder so that drift error due to losses can be greatly reduced. Specifically, such error-concealment techniques can be applied as long as the losses on the different paths are not concurrent. One advantage of the MD scheme is simplicity, since path selection is trivial, and compression can be performed independently of the network conditions. It should be noted that the joint reference frame and path selection subsumes the above MD example as a special case, at the expense of additional computation.

Unlike many previous rate-distortion optimization algorithms [7] [5] [8] which rely on the use of Lagrange multipliers, our optimization is unique in that we use an integer rounding technique that allows trade-off between computation complexity with the quality of the obtained solution. This allows us to estimate the quality of the obtained solution given fixed computation resources. Conversely, given a target quality of the solution, we can estimate the amount of resources needed for the tasks.

Among our previous work, we have shown that integer rounding-based complexity scaling can be applied to reference frame / QoS selection for uni-path streaming over QoS-enabled networks [9], and to reference frame / path selection for multi-path streaming over best-effort networks [10]. This paper is a noted improvement on our previous work in two important regards: i) we are simultaneously selecting reference frame, QoS and path for multi-path streaming over QoS-enabled networks; and, ii) in addition to the previously developed dynamic programming *dimension rounding* technique, to be discussed in Section V-A, a new rounding technique called *index rounding* is introduced in Section V-C, and the two types of rounding techniques are compared and combined in Section V-D. The new rounding technique will be shown to be superior in performance in Section VI-D.

III. Problem Formulation

Regardless of whether network QoS is obtained via network mechanisms such as DiffServ [2] or application-level mechanisms such as FEC, one important property is that different QoS service levels will likely lead to different packet loss rates, which is the primary network impairment we consider in this work. We begin with a discussion of the source and network models, in Section III-A and III-B, respectively. Given the assumed models, we discuss the objective function and formalize the optimization in Section III-C.

A. Directed Acyclic Graph Source Model

We assume an M-frame video sequence is coded as an intra-coded frame (I-frame) followed by $M-1$ inter-coded frames (P-frames). We model the decoding dependencies of the video using a directed acyclic graph (DAG) model $G=(\mathcal{V},\mathcal{E})$ with vertex set $\mathcal{V}, |\mathcal{V}|=M$ and edge set $\mathcal{E}$, similar to one used in [7]. Specifically, the streaming media is represented by a collection of frames, F_i's, $i \in \{1,\ldots,M\}$. Each frame F_i, represented by a node $i \in \mathcal{V}$, has a set of outgoing edges $e_{i,j} \in \mathcal{E}$ to nodes j's. Frame i can use frame j as reference iff $\exists e_{i,j} \in \mathcal{E}$. We define $x_{i,j}$ to be the binary variable indicating whether F_i uses F_j as reference. Or equivalently, given i, we define $x_{i,j}$ as:

$$x_{i,j} = \begin{cases} 1 & \text{if } F_i \text{ uses } F_j \text{ as RF} \\ 0 & \text{otherwise} \end{cases} \qquad \forall j \in \{\mathcal{V} | e_{i,j} \in \mathcal{E}\} \tag{1}$$

Because a P-frame can have only one reference frame, we have the following *RF constraint*:

$$\sum_{\{j | e_{i,j} \in \mathcal{E}\}} x_{i,j} = 1 \qquad \forall i \in \mathcal{V}, i \neq 1 \tag{2}$$

We assume that only frames in the past are used for reference, i.e. $\forall e_{i,j} \in \mathcal{E}, i > j$. Further, since in practice it is inefficient to use a reference frame too far in the past, we will limit the number of candidate reference frames for any given predicted frame F_i to be $E_{\max} \ll |\mathcal{V}|$. We also assume only frame 1 is intra-coded, and hence $\nexists e_{1,j} \in \mathcal{E}$. An example of a DAG model of a 4-frame sequence is shown in Figure 1 with $E_{\max}=2$.

We let $r_{i,j}$ be the integer number of bytes in frame i when frame j is used as reference. This is an approximation since the number of bytes depends not only on specific j chosen, but also the reference frame for frame j and so on. The byte size of the starting I-frame is $r_{1,1}$. We assume a sparse *rate matrix* $\mathbf{r}$ of size $O(M^2)$ is computed *a priori* as input to the optimization algorithm (sparse because each row has at

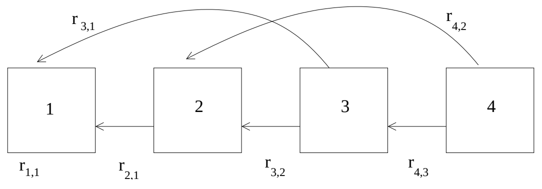

Fig. 1. Directed Acyclic Graph Source Model

most $E_{\max}$ entries). We will discuss how $\mathbf{r}$ is generated in our experiment in section VI.

B. Network Model

Regardless of whether QoS is established using network or application level mechanisms, we assume a set of QoS service levels $\mathcal{Q} = \{0, 1, \ldots Q\}$ to be available for all accessible transmission paths. We allow each frame F_i to select a QoS level $q_i \in \mathcal{Q}$ and a transmission path $t_i \in \{0, 1\}$. The special QoS service level $q_i = 0$ denotes the case when F_i is not transmitted at all. For given observable network condition, QoS level q_i, transmission path t_i and frame size $r_{i,j}$ will induce a frame delivery success probability $p_{t_i}(q_i, r_{i,j}) \in \mathcal{R}$, where $0 \leq p_{t_i}(q_i, r_{i,j}) \leq 1$. There is dependence of $p()$ on $r_{i,j}$ because a large frame size will likely negatively impact the delivery success probability of the entire frame as more data is pushed through the network. It should be noted beforehand that the operation and the optimality of our algorithm are independent of the form of $p_{t_i}(q_i, r_{i,j})$.

1) Network Resource Constraint: Like any resource allocation problems, we impose constraints on the amount of resource we can use, which in this case is the aggregate ability to protect the M-frame sequence per transmission path from network losses using QoS. Assuming a QoS assignment q_i results in a cost of $c(q_i) \in \mathcal{R}$ per byte, the constraints for path 0 and path 1 are respectively:

$$\sum_{i=1}^{M} \sum_{\{j|e_{i,j} \in \mathcal{E}\}} x_{i,j}\; c(q_i)\; (1 - t_i)\; r_{i,j} \quad \leq \quad \bar{R}_0$$
$$\sum_{i=1}^{M} \sum_{\{j|e_{i,j} \in \mathcal{E}\}} x_{i,j}\; c(q_i)\; t_i\; r_{i,j} \quad \leq \quad \bar{R}_1 \qquad (3)$$

In the case of network-level QoS, (3) represents a cost constraint per path, so that total cost to the user per M-frame time for path 0 and 1 do not exceed $\bar{R}_0$ and $\bar{R}_1$ monetary units, respectively, where $\bar{R}_0, \bar{R}_1 \in \mathcal{I}$. In the case of application-level QoS, (3) represents a bit rate constraint per path, where $c(q_i)$ is the overhead in channel coding given QoS level q_i, and constraint parameters $\bar{R}_0$ and $\bar{R}_1$ can be obtained using congestion control algorithms like [11], so that the total output bytes for M-frame time for path 0 and 1 do not exceed $\bar{R}_0$ and $\bar{R}_1$ bytes, respectively.

C. Objective Function

The objective function we selected is the expected number of correctly decoded frames at the decoder. Each frame F_i is correctly decoded iff F_i and all frames F_j's it depends on are delivered drop-free. We write $j \preceq i$ if frame i depends on frame j. Mathematically, maximizing the objective function means computing:

$$\max_{\{x_{i,j}\},\{q_i\},\{t_i\}} \left\{ \sum_{i=1}^{M} \prod_{j \preceq i} \sum_{\{k|e_{j,k} \in \mathcal{E}\}} x_{j,k}\; p_{t_j}(q_j, r_{j,k}) \right\} \qquad (4)$$

The problem is then: given pre-computed rate matrix $\mathbf{r}$, delivery success probability function $p_{t_i}(q_i, r_{i,j})$ and cost function $c(q_i)$, find variables $\{x_{i,j}\}$, $\{q_i\}$ and $\{t_i\}$ that maximize (4) while satisfying the integer constraint (1), the RF constraint (2) and the network resource constraints (3). This formally defined optimization is called the *RF / QoS / Path selection problem* (RQP selection).

IV. A Dynamic Programming Solution

Given the RQP selection problem is NP-hard (proof similar to one in [9]), we first present a pseudo-polynomial algorithm that solves the optimization problem optimally but in exponential time. Methodologies to tradeoff complexity with solution quality are discussed in later sections.

The optimization algorithm composes of two recursive functions, $Sum(i, R_0, R_1)$ and $Prod(j, i, R_0, R_1)$. $Sum(i, R_0, R_1)$ returns the maximum sum of products of sums in (4) for a subset of frames F_1 to F_i, given R_0 and R_1 network resource units are available for path 0 and 1, respectively. $Prod(j, i, R_0, R_1)$ returns the inner product of sums term in (4) for F_j — probability that F_j is *decoded* correctly — given R_0 and R_1 network resource units of path 0 and 1 are optimally distributed from F_1 to F_i. A call to $Sum(M, \bar{R}_0, \bar{R}_1)$ will yield the optimal solution. $Sum(i, R_0, R_1)$ and $Prod(j, i, R_0, R_1)$ are shown in pseudo-code in Figure 2 and 3, respectively. We now examine the pseudo-code closely.

A. Dissecting $Sum(i, R_0, R_1)$

The recursive case (line 10-19) is essentially testing every combination of RF, QoS and path for F_i for the maximum sum. The results of this search are stored in the $[i, R_0, R_1]$ entries of the four dynamic programming (DP) tables, $DPsum[\,]$, $DPqos[\,]$, $DPind[\,]$ and $DPpath[\,]$ (line 20-21). DP tables are used so that if the same subproblem is called again, the already computed result can be simply returned (line 1-2). The two base cases (line 3-9) are the following: i) when one or both of the resource constraints are violated, in which case we return $-\infty$ to signal the violation; and, ii) when the root node (I-frame) is reached. Because root node has no RF to choose from, the search for optimal solution (line 6-8) is much simpler.

Assuming $Prod(j, i, R_0, R_1)$ does not introduce further complexity (to be discussed), the complexity of $Sum(M, \bar{R}_0, \bar{R}_1)$ is bounded by the time required to construct the DP table of dimension $M * \bar{R}_0 * \bar{R}_1$. To fill each entry, we call function $Sum(i, R_0, R_1)$ as shown in Figure 2, which has $O(E_{\max} Q)$ operations to account for the two `for` loops from

```
function $Sum(i, R_0, R_1)$
1.  if  ( $DPsum[i, R_0, R_1]$ is filled )        // DP case
2.  {    return $DPsum[i, R_0, R_1]$;    }
3.  if  ( $R_0 < 0$ ) or ( $R_1 < 0$ )           // base case 1
4.  {    return $-\infty$;    }
5.  if  ( $i = 1$ )                           // base case 2
6.  {    $s_0 := \max_{q \in \{\mathcal{Q} | c(q) r_{1,1} \leq R_0\}} p_0(q, r_{1,1})$;
7.       $s_1 := \max_{q \in \{\mathcal{Q} | c(q) r_{1,1} \leq R_1\}} p_1(q, r_{1,1})$;
8.       return $\max(s_0, s_1)$;
9.  }
10. $S$ := 0;                                  // recursive case
11. for each $j$ such that $e_{i,j} \in \mathcal{E}$,
12. {    for each $q \in \mathcal{Q}$,
13.    { $s_0 := Sum(i-1, R_0 - c(q) r_{i,j}, R_1) +$
              $p_0(q, r_{i,j})\ Prod(j, i-1, R_0 - c(q) r_{i,j}, R_1)$;
14.      $s_1 := Sum(i-1, R_0, R_1 - c(q) r_{i,j}) +$
              $p_1(q, r_{i,j})\ Prod(j, i-1, R_0, R_1 - c(q) r_{i,j})$;
15.      $s := \max(s_0, s_1)$;
16.      $t := (s_0 > s_1)?\ \ 0 : 1$;
17.      if  ( $s > S$ )
18.      {    $(S, X, Y, Z) := (s, q, j, t)$;    }
19. } }
20. ( $DPsum[i, R_0, R_1]$, $DPqos[i, R_0, R_1]$ ) := ( $S, X$ );
21  ( $DPind[i, R_0, R_1]$, $DPpath[i, R_0, R_1]$ ) := ( $Y, Z$ );
22. return $S$;
```

Fig. 2. Defining $Sum(i, R_0, R_1)$

line 11-19 in the recursive case. Therefore we can conclude the complexity of $Sum(M, \bar{R}_0, \bar{R}_1)$ is $O(ME_{\max} Q \bar{R}_0 \bar{R}_1)$.

B. Dissecting $Prod(j, i, R_0, R_1)$

From line 13 and 14 of Figure 2, we assume that $Prod(j, i, R_0, R_1)$ is called after $Sum(i, R_0, R_1)$ has been called, so we will assume entry $[i, R_0, R_1]$ of the DP tables are available during execution of $Prod(j, i, R_0, R_1)$.

The recursive case has two sub-cases: i) when $j < i$ (line 6-11), in which case we recurse on $Prod(j, i-1, .)$ given we know resource $c(X) r_{i,Y}$ is optimally used for node i; and, ii) when $j = i$ (line 12-17), in which case we know term i of the product term — $p_0(X, r_{i,Y})$ if path 0 and $p_1(X, r_{i,Y})$ if path 1. The maximum product will be this term times the recursive term $Prod(Y, i-1, R_0 - c(X) r_{i,Y}, R_1)$ if path 0 and $Prod(Y, i-1, R_0, R_1 - c(X) r_{i,Y})$ if path 1. The two base cases (line 1-5) are similar to the two base cases for $Sum(i, R_0, R_1)$.

Though not written in the code in Figure 3 for simplicity of presentation, a DP table $DPprod[j][i][R_0][R_1]$ can be similarly used to store solutions to subproblems to avoid solving the same subproblem twice. Because the number of reference frames is bounded by $E_{\max}$, at most $E_{\max} * M * \bar{R}_0 * \bar{R}_1$ entries of the DP table will be filled. The complexity of $Prod(j, i, R_0, R_1)$ is also bounded by the time required to fill the $O(E_{\max} * M * \bar{R}_0 * \bar{R}_1)$ necessary entries of the DP table. Since there are no loops in Figure 3, we say it takes constant time to fill each entry in the DP table. Hence the complexity of $Prod(j, M, \bar{R}_0, \bar{R}_1), \forall j$ s.t. $E_{i,j} \in \mathcal{E}$, is $O(E_{\max} M \bar{R}_0 \bar{R}_1)$. The complexity of $Sum(M, \bar{R}_0, \bar{R}_1)$ dominates this complexity, hence the aggregate complexity of the algorithm is $O(ME_{\max} Q \bar{R}_0 \bar{R}_1)$.

```
function $Prod(j, i, R_0, R_1)$
1.  if  ( $R_0 < 0$ ) or ( $R_1 < 0$ )          // base case 1
2.  {  return 0;    }
3.  if  ( $j = i = 1$ )                        // base case 2
4.  {  return $DPsum[1, R_0, R_1]$;    }
5.  ( $X, Y$ ) := ( $DPqos[i, R_0, R_1]$,  $DPind[i, R_0, R_1]$ );
6.  if  ( $j < i$ )                          // recursive case
7.  { if  ( $DPpath[i, R_0, R_1] = 0$ )
8.     { $P := Prod(j, i-1, R_0 - c(X) r_{i,Y}, R_1)$;   }
9.     else
10.    { $P := Prod(j, i-1, R_0, R_1 - c(X) r_{i,Y})$;   }
11  }
12. else                                    // j = i
13. { if  ( $DPpath[i, R_0, R_1] = 0$ )
14.   { $P := p_0(X, r_{i,Y}) * Prod(Y, i-1, R_0 - c(X) r_{i,Y}, R_1)$;  }
15.     else
16.   { $P := p_1(X, r_{i,Y}) * Prod(Y, i-1, R_0, R_1 - c(X) r_{i,Y})$;  }
17. }
18. return $P$;
```

Fig. 3. Defining $Prod(j, i, R_0, R_1)$

V. Rounding-based Complexity Scaling

Maximizing objection function in (4) subject to constraints (3) is often too complex to be practical. In this section, we describe two mechanisms for constructing alternate constraints that allow trade-off between computation and solution quality. As we will see, the alternate constraints offer multiple trade-off points by adjusting a parameter, yield a feasible solution to the original constraints, and have approximation error that can be determined.

A. DP Dimension Rounding

As previously derived, the complexity of $Sum(M, \bar{R}_0, \bar{R}_1)$ is $O(ME_{\max} Q \bar{R}_0 \bar{R}_1)$, which is pseudo-polynomial [12]. This essentially means the complexity looks polynomial but is not. In this case, because $\bar{R}_0$ (similarly $\bar{R}_1$) is encoded in $\lceil \log_2 \bar{R}_0 \rceil$ bits as input, complexity $O(\bar{R}_0)$ means the algorithm is exponential in the size of the input parameters.

In practice, to scale down the algorithmic complexity when $\bar{R}_0$ and $\bar{R}_1$ are large, we perform *integer rounding-based complexity scaling*. The first rounding technique is *DP dimension rounding*. We first scale and round down overall budgets $\bar{R}_0$ and $\bar{R}_1$ by factor $K_{DR} \in \mathcal{R}$ — i.e. $\left\lfloor \frac{\bar{R}_0}{K_{DR}} \right\rfloor$ and $\left\lfloor \frac{\bar{R}_1}{K_{DR}} \right\rfloor$ — as input to the optimization. We then scale and round up costs of transmitting predicted frame F_i, $c(q_i) r_{i,j}$'s, by the same factor K_{DR} — i.e. $\left\lceil \frac{c(q_i) r_{i,j}}{K_{DR}} \right\rceil$. Implementationally, we accordingly rewrite line 13-14 of $Sum(i, R_0, R_1)$ of Figure 2:

$$13.\ s_0 := Sum\left(i-1, R_0 - \left\lceil \frac{c(q)\ r_{i,j}}{K_{DR}} \right\rceil, R_1\right) + p_0(q, r_{i,j})\ Prod\left(j, i-1, R_0 - \left\lceil \frac{c(q)\ r_{i,j}}{K_{DR}} \right\rceil, R_1\right);$$

$$14.\ s_1 := Sum\left(i-1, R_0, R_1 - \left\lceil \frac{c(q)\ r_{i,j}}{K_{DR}} \right\rceil\right) + p_1(q, r_{i,j})\ Prod\left(j, i-1, R_0, R_1 - \left\lceil \frac{c(q)\ r_{i,j}}{K_{DR}} \right\rceil\right);$$

Similarly, we replace the cost terms in $Prod(j, i, R_0, R_1)$ of Figure 3 by rewriting line 8, 10, 14 and 16:

$$8.\ \ P := Prod\left(j, i-1, R_0 - \left\lceil \frac{c(X)\ r_{i,Y}}{K_{DR}} \right\rceil, R_1\right);$$

$$10.\ P := Prod\left(j, i-1, R_0, R_1 - \left\lceil \frac{c(X)\ r_{i,Y}}{K_{DR}} \right\rceil\right);$$

```
14. $P := p_0(X, r_{i,Y}) * Prod(Y, i-1, R_0 - \left\lceil \frac{c(X)\ r_{i,Y}}{K_{DR}} \right\rceil, R_1)$;
16. $P := p_1(X, r_{i,Y}) * Prod(Y, i-1, R_0, R_1 - \left\lceil \frac{c(X)\ r_{i,Y}}{K_{DR}} \right\rceil)$;
```

In so doing, instead of solving the original RQP selection instance I for true optimal solution s^*, we solve an approximate instance I^A for solution s^A. Scaling down $\bar{R}_0$ and $\bar{R}_1$ means scaling down the dimension of the dynamic programming table, hence the complexity is reduced by a factor of K_{DR}^2 at the cost of decreasing solution quality. Using $Sum(i, R_0, R_1)$ and $Prod(j, i, R_0, R_1)$ with the rewritten lines, the complexity of I^A is now $O(ME_{\max}Q\bar{R}_0\bar{R}_1K_{DR}^{-2})$.

Note that in the approximate instance I^A, the network resource constraints become:

$$\sum_{i=1}^{M} \sum_{\{j|e_{i,j}\in\mathcal{E}\}} x_{i,j}\ (1-t_i) \left\lceil \frac{c(q_i)\ r_{i,j}}{K_{DR}} \right\rceil \leq \left\lfloor \frac{\bar{R}_0}{K_{DR}} \right\rfloor$$
$$\sum_{i=1}^{M} \sum_{\{j|e_{i,j}\in\mathcal{E}\}} x_{i,j}\ t_i \left\lceil \frac{c(q_i)\ r_{i,j}}{K_{DR}} \right\rceil \leq \left\lfloor \frac{\bar{R}_1}{K_{DR}} \right\rfloor \quad (5)$$

It is shown in the Appendix that s^A is feasible in I. Moreover, we can bound the performance difference between s^A and true optimal s^* by first obtaining a super-optimal solution s^S in a new problem instance I^S, where we replace $\bar{R}_0$ and $\bar{R}_1$ with $\left\lceil \frac{\bar{R}_0}{K_{DR}} \right\rceil$ and $\left\lceil \frac{\bar{R}_1}{K_{DR}} \right\rceil$, and replace $c(q_i)r_{i,j}$'s with $\left\lfloor \frac{c(q_i)r_{i,j}}{K_{DR}} \right\rfloor$. The super-optimal network resource constraints are:

$$\sum_{i=1}^{M} \sum_{\{j|e_{i,j}\in\mathcal{E}\}} x_{i,j}\ (1-t_i) \left\lfloor \frac{c(q_i)\ r_{i,j}}{K_{DR}} \right\rfloor \leq \left\lceil \frac{\bar{R}_0}{K_{DR}} \right\rceil$$
$$\sum_{i=1}^{M} \sum_{\{j|e_{i,j}\in\mathcal{E}\}} x_{i,j}\ t_i \left\lfloor \frac{c(q_i)\ r_{i,j}}{K_{DR}} \right\rfloor \leq \left\lceil \frac{\bar{R}_1}{K_{DR}} \right\rceil \quad (6)$$

After obtaining optimal solution s^S to I^S, we can bound our approximate solution s^A from the optimal s^* in original problem instance I as follows:

$$\left|\text{obj}(s^*) - \text{obj}(s^A)\right| \leq \left|\text{obj}(s^S) - \text{obj}(s^A)\right| \quad (7)$$

where obj(s) is the objective function (4) using solution s. The proof of performance bound (7) is also found in the Appendix.

B. Comparing Versions of DP Dimension Rounding

We realize that instead of scaling and rounding up the costs $c(q_i)r_{i,j}$'s by K_{DR}, it is possible to scale and round up only the rates $r_{i,j}$'s by K_{DR}, i.e. $\left\lceil \frac{r_{i,j}}{K_{DR}} \right\rceil$. The network constraints will then become:

$$\sum_{i=1}^{M} \sum_{\{j|e_{i,j}\in\mathcal{E}\}} x_{i,j}\ c(q_i)\ (1-t_i) \left\lceil \frac{r_{i,j}}{K_{DR}} \right\rceil \leq \left\lfloor \frac{\bar{R}_0}{K_{DR}} \right\rfloor$$
$$\sum_{i=1}^{M} \sum_{\{j|e_{i,j}\in\mathcal{E}\}} x_{i,j}\ c(q_i)\ t_i \left\lceil \frac{r_{i,j}}{K_{DR}} \right\rceil \leq \left\lfloor \frac{\bar{R}_1}{K_{DR}} \right\rfloor \quad (8)$$

Like the scaling and rounding down of $\bar{R}_0$ and $\bar{R}_1$ by factor K_{DR}, replacement of $r_{i,j}$'s can be done as a pre-processing step to modify input prior to optimization, and as a result, optimization algorithm $Sum(i, R_0, R_1)$ and $Prod(j, i, R_0, R_1)$ of Figure 2 and 3 can be reused unaltered for the complexity-scaled version as well. This is how DP dimension rounding was performed in our earlier work [9].

However, the current version of DP dimension rounding offered two advantages. First, by pulling cost term $c(q_i)$ inside the rounding function, cost function $c(q_i)$ can be real numbers, while the previous version requires $c(q_i), q_i \in \mathcal{Q}$, to be integers in order for $R_{t_i} - c(q_i)\left\lceil \frac{r_{i,j}}{K_{DR}} \right\rceil, t_i \in \{0,1\}$, to be integers and indexable to DP tables. Second, the current version has smaller round-off error: the largest possible left side rounding error of the current version's network constraint (5) is K_{DR} per term, which is smaller than the error of the previous version's (8), $c_{\max}K_{DR}, c_{\max} = \max_{q_i\in\mathcal{Q}}\{c(q_i)\}$, given $C_{\max} > 1$, $C_{\max} \in \mathcal{I}$. Since the current version is more general and more accurate, we prefer the current version of DP dimension rounding.

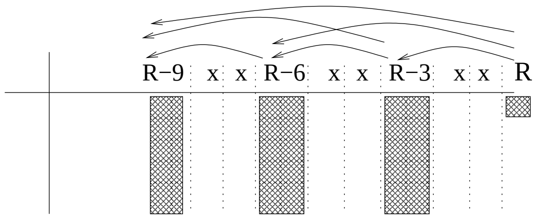

Fig. 4. Illustration of DP Index Rounding

C. DP Index Rounding

Instead of reducing the overall dimension of the dynamic programming table to scale down algorithmic complexity, another way is to limit the number of indices used in the DP table given the table dimension. This rounding technique is called *DP index rounding*, and we accomplish that by always subtracting a positive integer multiple of $K_{IR} \in \mathcal{I}$ from R_0 or R_1 during recursive calls in $Sum(i, R_0, R_1)$ of Figure 2. Implementationally, we do that by replacing $c(q_i)r_{i,j}$ with an approximate $K_{IR}\left\lceil \frac{c(q_i)r_{i,j}}{K_{IR}} \right\rceil$. Rewrite line 13-14 of $Sum(i, R_0, R_1)$, we get:

```
13. $s_0 := Sum(i-1, R_0 - K_{IR}\left\lceil \frac{c(q)r_{i,j}}{K_{IR}} \right\rceil, R_1) +$
        $p_0(q, r_{i,j})\ Prod(j, i-1, R_0 - K_{IR}\left\lceil \frac{c(q)r_{i,j}}{K_{IR}} \right\rceil, R_1)$;
14. $s_1 := Sum(i-1, R_0, R_1 - K_{IR}\left\lceil \frac{c(q)r_{i,j}}{K_{IR}} \right\rceil) +$
        $p_1(q, r_{i,j})\ Prod(j, i-1, R_0, R_1 - K_{IR}\left\lceil \frac{c(q)r_{i,j}}{K_{IR}} \right\rceil)$;
```

Similarly, we replace the cost terms in $Prod(j, i, R_0, R_1)$ of Figure 3 by rewriting line 8, 10, 14 and 16:

```
8.  $P := Prod(j, i-1, R_0 - K_{IR}\left\lceil \frac{c(X)\ r_{i,Y}}{K_{IR}} \right\rceil, R_1)$;
10. $P := Prod(j, i-1, R_0, R_1 - K_{IR}\left\lceil \frac{c(X)\ r_{i,Y}}{K_{IR}} \right\rceil)$;
14. $P := p_0(X, r_{i,Y}) * Prod(Y, i-1, R_0 - K_{IR}\left\lceil \frac{c(X)r_{i,Y}}{K_{IR}} \right\rceil, R_1)$;
16. $P := p_1(X, r_{i,Y}) * Prod(Y, i-1, R_0, R_1 - K_{IR}\left\lceil \frac{c(X)r_{i,Y}}{K_{IR}} \right\rceil)$;
```

As an example, we see an illustration of DP index rounding in Figure 4 when $K_{IR} = 3$. By recursing only on R less multiples of 3, we are only filling at most 1/3 of all indices along both R_0 and R_1 dimensions. Hence the new algorithmic complexity is: $O(ME_{\max}Q\bar{R}_0\bar{R}_1K_{IR}^{-2})$.

The new network constraints using DP index rounding are as follows:

$$\sum_{i=1}^{M}\sum_{\{j|e_{i,j}\in\mathcal{E}\}} x_{i,j}\ (1-t_i)\ K_{IR}\left\lceil\frac{c(q_i)\ r_{i,j}}{K_{IR}}\right\rceil \leq \bar{R}_0$$
$$\sum_{i=1}^{M}\sum_{\{j|e_{i,j}\in\mathcal{E}\}} x_{i,j}\ t_i\ K_{IR}\left\lceil\frac{c(q_i)\ r_{i,j}}{K_{IR}}\right\rceil \leq \bar{R}_1 \qquad (9)$$

Using similar opposite rounding technique in the previous section, we can bound the performance of the approximate solution from the optimal by first constructing a super-optimal solution s^S and evaluating bound (7). Proof will be similar to the one in the DP dimension rounding case (shown in the Appendix) and hence is omitted here.

D. Applying DP Dimension & Index Rounding

We can employ both rounding strategies simultaneously: replace $\bar{R}_0$ and $\bar{R}_1$ with $\left\lfloor\frac{\bar{R}_0}{K_{DR}}\right\rfloor$ and $\left\lfloor\frac{\bar{R}_1}{K_{DR}}\right\rfloor$ respectively as input to the algorithm, and replace $c(q)r_{i,j}$ with $K_{IR}\left\lceil\frac{c(q)r_{i,j}}{K_{IR}K_{DR}}\right\rceil$ in line 13-14 of recursive function $Sum(i,R_0,R_1)$ of Figure 2 and line 8, 10, 14 and 16 of $Prod(j,i,R_0,R_1)$ of Figure 3. The resulting network constraints are:

$$\sum_{i=1}^{M}\sum_{\{j|e_{i,j}\in\mathcal{E}\}} x_{i,j}\ (1-t_i)\ K_{IR}\left\lceil\frac{c(q_i)\ r_{i,j}}{K_{IR}K_{DR}}\right\rceil \leq \left\lfloor\frac{\bar{R}_0}{K_{DR}}\right\rfloor$$
$$\sum_{i=1}^{M}\sum_{\{j|e_{i,j}\in\mathcal{E}\}} x_{i,j}\ t_i\ K_{IR}\left\lceil\frac{c(q_i)\ r_{i,j}}{K_{IR}K_{DR}}\right\rceil \leq \left\lfloor\frac{\bar{R}_1}{K_{DR}}\right\rfloor \qquad (10)$$

The resulting complexity is $O(ME_{\max}Q\bar{R}_0\bar{R}_1K_{DR}^{-2}K_{IR}^{-2})$. An interesting question is then: given a desired complexity reduction factor K^2, where $K = K_{DR}K_{IR}$, what are the trade-offs in using different K_{DR} and K_{IR}?

Because our approximation bound (7) is an *a posteriori* bound instead of an *a priori* one, i.e. we do not know precisely the extent of the error until approximate solution s^A and super-optimal solution s^S are computed and evaluated, we cannot directly relate the performance of our approximate solution s^A to K_{DR} and K_{IR} analytically. To estimate the performance of the to-be-constructed approximate solution s^A *a priori* given rounding factors K_{DR} and K_{IR}, we instead focus on an alternate performance metric Ω_{err} that tracts the maximum possible rounding error to occur when calculating network resource constraints (10) instead of the original (3). In the worst case, Ω_{err} is the maximum rounding error on the right side of (10) plus the maximum rounding error on the left side. Right maximum error is the maximum rounding error between $\bar{R}_0$ and $\bar{R}_1$; left maximum error is the number of P-frames $(M-1)$ times the maximum rounding error of $c(q)r_{i,j}$:

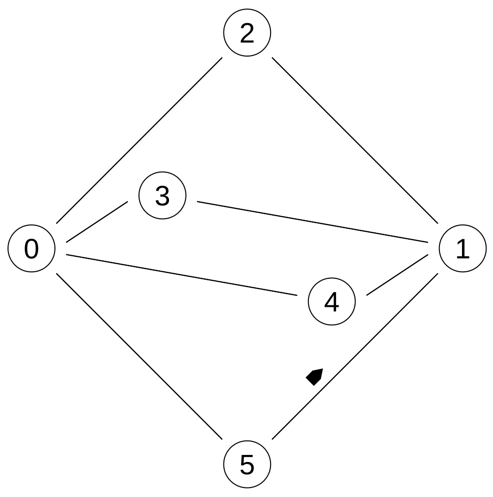

Fig. 5. Snapshot of NAM: Network Graph

$$\begin{aligned}\Omega_{err} &= \max\left\{\left|\bar{R}_0 - K_{DR}\left\lfloor\frac{\bar{R}_0}{K_{DR}}\right\rfloor\right|,\ \left|\bar{R}_1 - K_{DR}\left\lfloor\frac{\bar{R}_1}{K_{DR}}\right\rfloor\right|\right\} + \\ &\quad (M-1) * \max_{q_i,r_{i,j}}\left|c(q_i)\ r_{i,j} - K_{IR}K_{DR}\left\lceil\frac{c(q_i)r_{i,j}}{K_{IR}K_{DR}}\right\rceil\right| \\ &= K_{DR} + (M-1)K_{IR}K_{DR} \qquad (11)\end{aligned}$$

If we now substitute $K_{IR} = K/K_{DR}$ into (11), we get:

$$\Omega_{err} = K_{DR} + (M-1)K \qquad (12)$$

Hence Ω_{err} is a linear increasing function of K_{DR}, i.e. we should let $K_{IR} = K$ to minimize Ω_{err} for fixed K. Depending on implementation, in practice, we may need to use a larger K_{DR} to reduce the amount of memory needed for the DP tables, each of dimension $O(ME_{\max}Q\bar{R}_0\bar{R}_1K_{DR}^{-2})$. So a practical rounding factor selection strategy to achieve a complexity scaling factor of K^2, $K = K_{DR}K_{IR}$, is:

1) Select the smallest $K_{DR} \in \mathcal{R}$ that sufficient memory can be allocated for DP tables.
2) Given K and K_{DR}, calculate $K_{IR} := \left\lceil\frac{K}{K_{DR}}\right\rceil$.

VI. Experimentation

A. Experiment Setup

To test the optimization algorithm developed in Section IV, we built an experimental testbed using network simulator 2 (ns2) [13]. For the network, we use a simple network graph, shown as a snap shot of network animator (NAM) in Figure 5. First, we constructed a node each for both the streaming server (node 0) and the client (node 1). To simulate different network path packet loss characteristics, we constructed four application-level forwarding agents (node 2-5), simulating the role of application nodes in a content delivery network, whose sole purpose is to forward packets to the client upon packet arrival from the server. Different network packet losses were simulated using an independent loss model of varying loss parameter values on the links from the forwarding agents to the client. This resulted in four paths of different packet loss rates.

We did not implement levels of quality-of-service (QoS) at the network layer in the experiment. Instead, we manually set the loss rate of the second path to be the effective packet loss rate if Reed-Solomon $RS(3,2)$ — recall that FEC schemes based on Reed-Solomon codes can correct as many losses as the number of redundancy packets [14] — is used over a channel with raw packet loss rate of the first path. We then manually enforced a combined rate constraint $\bar{R}_0$ for both first and second path. This is essentially equivalent to a single *virtual path* with one constraint and two QoS classes. We applied the same methodology to the third and fourth path create a second virtual path with rate constraint $\bar{R}_1$ and two QoS classes. We performed all our experiments using this two-virtual paths, two-QoS-level network graph.

At the application side, we use H.264 version JM4.2 [1] to encode the 100-frame QCIF (176×144) `news` sequence, sub-sampled in time by 2, at quantization parameter 31 and 30 for I-frames and P-frames, respectively. I-frame frequency is once every ten frames, and we optimized 10 frames at a time, i.e. one I-frame plus nine P-frames. To generate rates $r_{i,j}$'s as input to the optimization algorithm, we iteratively force each predicted frame F_i to use reference frame F_{i-t} for motion prediction during iteration $t = \{1,2,3,4,5\}$. The resulting coding rate is $r_{i,i-t}$. We assume a predicted frame F_i will use a reference frame no further back in time than F_{i-5}, or simply $E_{\max} = 5$. The resulting $r_{i,j}$'s are loaded into ns2 as a preprocessing step prior to optimization.

B. Numerical Results 1: RQP Selection Comparison

For the first set of experiment, we show that our optimization, as a streaming optimization scheme, has practical merits and out-performs a competing ad-hoc scheme. As previously mentioned, we assume two virtual paths each with two levels of QoS: regular packet transmission, and forward error correction $RS(3,2)$ protected. First, we fixed the combined bandwidth of the two virtual paths, $\bar{R}_0 + \bar{R}_1$, at $55kps$. Rounding parameters K_{DR} and K_{IR} were constant at 100 and 1, respectively. By varying the share of $55kbps$ bandwidth to the first virtual path bandwidth $\bar{R}_0$, we tracked the corresponding performance at the client in percentage of correctly decoded frames. A frame F_i is deemed correctly decoded iff F_i is correctly delivered *and* all its dependent frames are correctly decoded. The sequence was replayed 1200 times for an averaging effect. Two trials of different packet loss rates of the two virtual paths were performed: $(0.06, 0.10)$ and $(0.04, 0.08)$.

We compared our optimization scheme to an ad-hoc scheme we call `MD-greedy`: to choose reference frames, it divides the frames into groups of even frames and odd frames so that frame F_i uses F_{i-2} as reference, with the exception of F_1 which uses F_0, and F_0 which is an I-frame and has no reference frame. This essentially creates multiple (two) *descriptions* of the video that are independently decodeable (with the exception of odd frames' dependency on F_0). To

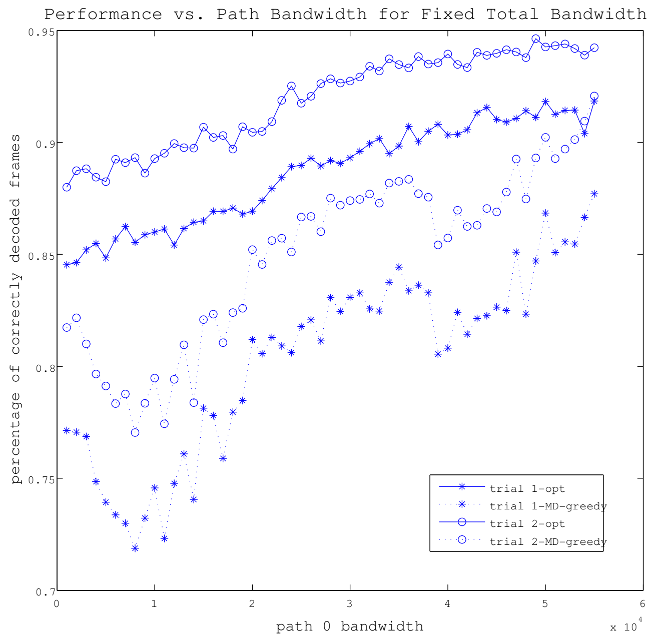

Fig. 6. Streaming Performance for varying Path 1 bandwidth for fixed total bandwidth. Packet loss rates for paths (1, 2) in trials 1 and 2 are (6%, 10%) and (4%, 8%), respectively.

choose transmission path, even frames will use virtual path 0 and odd frames will use virtual path 1. If the rate budgets $\bar{R}_0$ and $\bar{R}_1$ do not permit this path selection, then frames at the end of the dependency chains will be switched to the other path until the budget is met. To choose level of QoS, `MD-greedy` uses a water-filling method that raises the QoS level of the frames in the front of the dependency chains as much as possible until the budgets $\bar{R}_0$ and $\bar{R}_1$ are empty.

The performance as a function of the first virtual path bandwidth $\bar{R}_0$ is shown in Figure 6. First, we see that the two performance curves of `MD-greedy` is essentially a shifted version of the other. This is expected, as `MD-greedy`'s RQP selection depends solely on the bandwidth budget and not network loss rates. On the other hand, at a particular network setting `opt` dynamically performs RQP selection based on all observable factors including bandwidth budgets and network loss rates. Hence the two performance curves of `opt` have distinct shapes. Second, we see that our optimization `opt` always outperformed the ad-hoc scheme `MD-greedy`: in the first trial, `opt` outperformed `MD-greedy` by 3.74% to 13.82%, and the second trial, by 2.16% to 12.28%. In both trials, we see that the performance increased as $\bar{R}_0$ increased. This is intuitive since the raw packet loss rate in both trials was smaller for virtual path 0 than path 1.

C. Numerical Results 2: Performance / Complexity Tradeoff

In the second experiment, we held the bandwidth of the two virtual paths $\bar{R}_0$ and $\bar{R}_1$ constant at $20kps$ and $35kbps$, respectively, DP index rounding parameter K_{IR} constant at 1, and varied K_{DR} to observe the tradeoff between performance

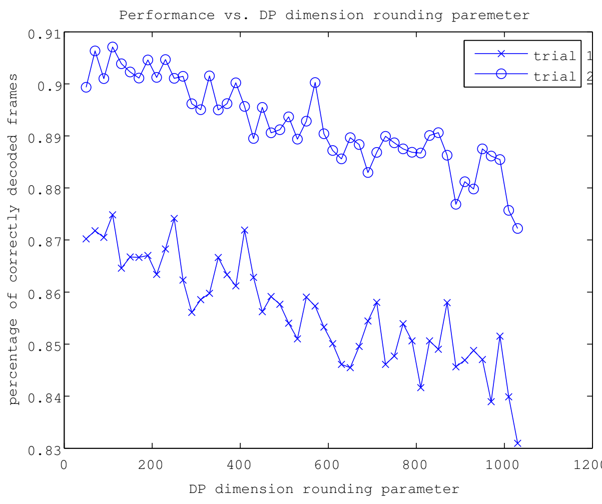

Fig. 7. Quality degradation as dimension rounding parameter (K_{DR}) increases.

and complexity; recall the complexity of the optimization is $O(ME_{\max}Q\bar{R}_0\bar{R}_1K_{DR}^{-2}K_{IR}^{-2})$. Again, we performed two trials of different packet loss rates of the two virtual paths: $(0.06, 0.10)$ and $(0.04, 0.08)$. The performance as a function of K_{DR} for both trials can be seen in Figure 7.

We see in Figure 7 that indeed as DP dimension rounding factor K_{DR} increased, the quality of the solution suffered due to rounding and the performance decreased for both trials. What is more interesting is that the curves are not stepwise strictly decreasing. This can be attributed to the fact that rounding is a non-linear operation, meaning that the precise degree of the rounding error will depend on actual numbers $r_{i,j}$'s, $\bar{R}_0$ and $\bar{R}_1$ as well as K_{DR}. The general downward trend of the curves, however, is in agreement with our analysis in Section V-A that performance is in general inversely proportional to rounding factor K_{DR}.

D. Numerical Results 3: Dimension Rounding vs. Index Rounding

In the third experiment, we seek to determine the merits of DP index rounding as opposed to DP dimension rounding for given algorithmic complexity. We held the loss rate and bandwidth of the two virtual paths constant at $(0.06, 0.1)$ and $(30kbps, 35kbps)$, respectively. We varied K_{IR} but kept the complexity reduction factor $K = K_{IR} * K_{DR}$ constant by adjusting $K_{DR} = K/K_{IR}$. Recall that K_{IR} must be an integer while K_{DR} needs not be. The result plot for three different values of K is shown in Figure 8.

We first note in Figure 8 that as the complexity reduction factor K decreased, the performance curve correspondingly moved up — i.e. performance increased. This agrees with our observation in previous part 2 of the experiment that performance is inversely proportional to complexity reduction factor. Second, in all three cases, as DP index rounding factor K_{IR} increased for fixed K, the performance increased. This agrees with our analytical conclusion in Section V-D

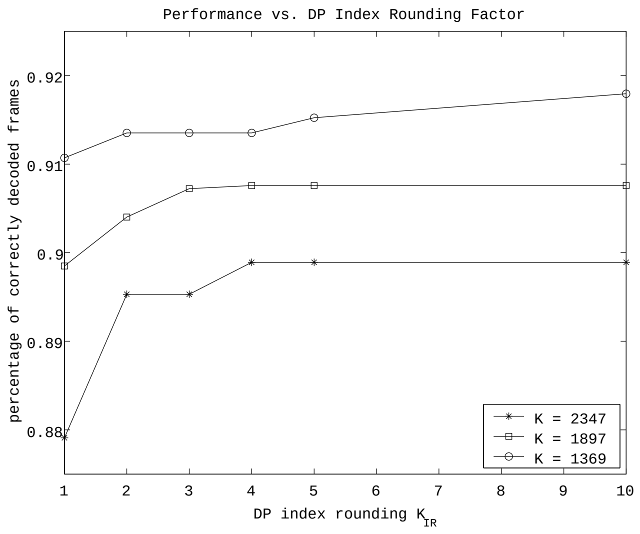

Fig. 8. Comparison of Index Rounding and Dimension Rounding for different values of $K = K_{IR}K_{DR}$.

that because K_{IR} has smaller maximum error than K_{DR}, in general it is better to use K_{IR} as opposed to K_{DR} to reduce complexity. We do observe, however, that the increase in performance of using K_{IR} as oppose to K_{DR} level off and no further improvement can be seen beyond a certain point.

VII. Conclusion

In this paper, we consider optimized streaming for H.264 video over multiple QoS levels and multiple transmission paths of different loss characteristics. In particular, we presented an optimization scheme that maximizes the expected number of correctly decoded frames at the receiver by selecting the near-optimal reference frame, QoS level and transmission path for each predicted frame. The optimization is novel in the sense that unlike convention Lagrangian approaches, it uses a mixture of two rounding techniques, DP dimension rounding and DP index rounding, to gracefully trade off complexity with the quality of the obtained solution. Experiments show improvement over a static reference frame selection scheme.

Appendix

To prove feasibility of approximate solution s^A in I and the performance bound (7), we essentially need to prove two axioms: i) that s^A satisfies original network constraints (3), ii) that s satisfies super-optimal network constraints (6) in I^S. To prove the first axiom, we first let the approximate solution be $s^A = \left(\{x^A_{i,j}\}, \{q^A_i\}, \{t^A_i\}\right)$. Given s^A satisfies the first network constraint in (5), we can write:

$$\sum_{i=1}^{M} \sum_{\{j|e_{i,j}\in\mathcal{E}\}} x^A_{i,j}(1-t^A_i)\left\lceil \frac{c(q^A_i)\; r_{i,j}}{K_{DR}} \right\rceil K_{DR} \;\le\; \left\lfloor \frac{\bar{R}_0}{K_{DR}} \right\rfloor K_{DR}$$

$$\sum_{i=1}^{M} \sum_{\{j|e_{i,j}\in\mathcal{E}\}} x^A_{i,j}(1-t^A_i)\frac{c(q^A_i)\; r_{i,j}}{K_{DR}} K_{DR} \;\le\; \frac{\bar{R}_0}{K_{DR}} K_{DR} \quad (13)$$

where (13) holds since $\frac{c(q_i)r_{i,j}}{K_{DR}} \leq \left\lceil \frac{c(q_i)r_{i,j}}{K_{DR}} \right\rceil$ and $\frac{R_t}{K_{DR}} \geq \left\lfloor \frac{R_t}{K_{DR}} \right\rfloor$. Similar steps can be done for the second network constraint. Therefore the first axiom holds and s^A is feasible in I. Using similar argument, one can show easily the second axiom: that s is feasible in I^S. By optimality of s^S in solution space of I^S, we have:

$$\text{obj}(s^S) \geq \text{obj}(s) \tag{14}$$

By subtracting obj(s^A) and taking absolute value on both sides, we get (7).

References

[1] "The TML project web-page and archive," http://kbc.cs.tu-berlin.de/ stewe/vceg/.

[2] S. Blake *et al.*, "An architecture for differentiated services," December 1998, IETF RFC 2475.

[3] G. Montenegro, S. Dawkins, M. Kojo, V. Magret, and N. Vaidya, "Long thin networks," January 2000, IETF RFC 2757.

[4] T. Wiegand, N. Farber, and B. Girod, "Error-resilient video transmission using long-term memory motion-compensated prediction," in *IEEE J. Select. Areas. Comm.*, vol. 18, no.6, June 2002, pp. 1050–1062.

[5] Y. Liang, M. Flieri, and B. Girod, "Low-latency video transmission over lossy packet networks using rate-distortion optimized reference picture selection," in *IEEE International Conference on Image Processing*, Rochester, NY, September 2002.

[6] J. Apostolopoulos, "Error-resilient video compression via multiple state streams," *Proc. International Workshop on Very Low Bitrate Video Coding (VLBV'99)*, pp. 168–171, October 1999.

[7] P. Chou and Z. Miao, "Rate-distortion optimized streaming of packetized media," Microsoft Research, Tech. Rep. MSR-TR-2001-35, February 2001.

[8] Y. Liang, E. Setton, and B. Girod, "Channel-adaptive video streaming using packet path diversity and rate-distortion optimized reference picture selection," in *IEEE Workshop on Multimedia Signal Processing*, St. Thomas, US Virgin Islands, December 2002.

[9] G. Cheung and C. Chan, "Jointly optimal reference frame & quality of service selection for h.26l video coding over lossy networks," in *IEEE International Conference on Multimedia and Expo*, Baltimore, MD, July 2003.

[10] G. Cheung, "Near-optimal multipath streaming of h.264 using reference frame selection," in *IEEE International Conference on Image Processing*, Barcelona, Spain, September 2003.

[11] S. Floyd, M. Handley, J. Padhye, and J. Widmer, "Equation-based congestion control for unicast applications," in *ACM SIGCOMM*, Stockholm, Sweden, August 2000.

[12] M. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. Freeman, 1979.

[13] "The network simulator ns-2," August 2003, release 2.26, http://www.isi.edu/nsnam/ns/.

[14] P. Frossard and O. Verscheure, "Joint source/FEC rate selection for quality-optimal MPEG-2 video delivery," in *IEEE Trans. Image Processing*, vol. 10, no.12, December 2001, pp. 1815–1825.