# Coding Structure and Replication Optimization for Interactive Multiview Video Streaming

Dongni Ren, S.-H. Gary Chan, *Senior Member, IEEE*, Gene Cheung, *Senior Member, IEEE*, and Pascal Frossard, *Senior Member, IEEE*

***Abstract*—Multiview video refers to videos of the same dynamic 3-D scene captured simultaneously by multiple closely spaced cameras from different viewpoints. We study interactive streaming of pre-encoded multiview videos, where, at any time, a client can request any one of many captured views for playback. Moreover, the client can periodically freeze the video in time and switch to neighboring views for a compelling look-around visual effect. We consider distributed content servers to support large-scale interactive multiview video service. These servers collaboratively replicate and access video contents. We study two challenges in this setting: what is an efficient coding structure that supports interactive view switching and, given that, what to replicate in each server in order to minimize the cost incurred by interactive temporal and view switches? We first propose a redundant coding structure that facilitates interactive view-switching, trading off storage with transmission rate. Using the coding structure, we next propose a content replication strategy that takes advantage of *indirect hit* to lower view-switching cost: in the event that the exact requested view is not available locally, the local server can fetch a different but correlated view from the other servers, so that the remote repository only needs to supply the pre-encoded view differential. We formulate the video content replication problem to minimize the switching cost as an integer linear programming (ILP) problem and show that it is NP-hard. We first propose an LP relaxation and rounding algorithm (termed Minimum Eviction) with bounded approximation error. We then study a more scalable solution based on dynamic programming and Lagrangian optimization (DPLO) with little sacrifice in performance. Simulation results show that our replication algorithms achieve substantially lower switching cost compared to other content replication schemes.**

***Index Terms*—Multimedia computing, digital video broadcasting.**

Manuscript received July 04, 2013; revised December 19, 2013; accepted June 09, 2014. Date of publication June 20, 2014; date of current version October 13, 2014. This work was supported in part by Hong Kong Research Grant Council (RGC) General Research Fund under Grant 610713, HKUST under Grant FSGRF13EG15, and the Hong Kong Innovation and Technology Fund under Grant UIM/246. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Ebroul Izquierdo.

S.-H. Gary Chan and D. Ren are with the Hong Kong University of Science and Technology, Clear Water Bay, Hong Kong (e-mail: tonyren@cse.ust.hk; gchan@cse.ust.hk).

G. Cheung is with the National Institute of Informatics, Tokyo 1018430, Japan (e-mail: cheung@nii.ac.jp).

P. Frossard is with the Ecole Polytechnique Fédérale de Lausanne (EPFL) Lausanne CH-1015, Switzerland (e-mail: pascal.frossard@epfl.ch).

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.2014.2332139

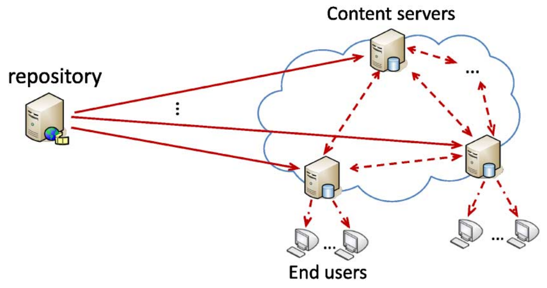


Fig. 1. Distributed interactive multiview video streaming (IMVS) network.

## I. Introduction

A MULTIVIEW video is a set of 2-D image sequences capturing the same dynamic 3-D scene recorded by a large array of time-synchronized and closely spaced cameras from $U$ different viewpoints [1]. We consider interactive streaming of multiview video, called *interactive multiview video streaming* (IMVS) in the sequel, where a client can freely select any one of these $U$ stored views and play back on a conventional 2-D display. In addition, the client can freeze the video in time and switch to nearby viewpoints to examine the 3-D scene from different angles [2]. This *static view-switching* interaction, which enables a "Matrix"-like look-around visual effect[1] with the action scene frozen in time, has been shown to be more appealing to users than *dynamic view-switching*, where the video is played back in time uninterrupted as a viewer switches to neighboring views, resembling a single-camera panning effect [3].

In order to support a large-scale service, a content provider often deploys content servers close to user pools [4]. We show in Fig. 1 an example of such a network. There is a remote repository storing all the pre-encoded videos. The content servers then collaboratively replicate the videos subject to their storage capacities. A client is served by its local server. In addition to temporal interactive requests (random access in time of the same video view using traditional DVR functionalities), a client can send inter-view requests (aforementioned static view-switching) to its local server. The server fulfills these requests directly if the data has been replicated locally. This situation is called *replication hit*. Otherwise, it contacts the other servers for the missing data. If no other servers has the requested data (a *replication miss*), the remote repository supplies it to the local server. While the network transmission delay

[1]During the filming of the 1999 movie "The Matrix", a 1D array of closely spaced cameras were arranged in a near half-circle and used to capture an action scene simultaneously. Frames of the same captured time instant from consecutive views were then arranged together for playback, enabling a look-around visual observation of the scene frozen in time.

1520-9210 © 2014 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission.
See http://www.ieee.org/publications_standards/publications/rights/index.html for more information.

is negligible among the servers, the delay from the repository is generally larger—potentially resulting in video playback delay—and the transmission is more expensive. Hence it is important to limit repository access as much as possible.

During a temporal or inter-view switching, the client has deliberately chosen a different navigation path, and a typical video client has to empty its current buffer content and rebuffer a fixed number of new frames before playback resumes. The delay in resuming video playback adversely affects viewing experience. Thus, we define *switching cost* to be the amount of time required for the rebuffering of the newly transmitted frames, which depends on both the size of the frames and the network location from which the frames are fetched.

In this paper, we study two critical challenges to support large-scale IMVS services: how to design an efficient coding structure to facilitate interactive view-switching? Then, given such a coding structure, how to optimize content replication across servers for multiple movies and minimize switching cost during interactive temporal and static view switches?

First, the multiview video must be pre-encoded *a priori* without knowing what view navigation paths the clients will eventually choose; these pre-encoded frames must be efficiently extracted for decoding and playback according to the actual viewpoints chosen by the clients at stream time. A simple (but naïve) way to enable random access for view-switching is to encode different viewpoints as independently coded Intra-frames (I-frames). In this case, selected views can be transmitted and decoded in any order since there is no coding dependency between views. However, because I-frames are not coding-efficient, this naïve solution results in unacceptably high bandwidth requirements.

Due to the high correlation among adjacent viewpoint images, we propose a more efficient coding structure to facilitate static view-switching, trading off required storage size with expected transmission rate. The structure is based on redundant P-frames [2] and *merge* frames [5] to help users navigate among adjacent views. A strength of our redundant coding structure is that, in addition to the replication hits (called *direct hits* in the sequel) and replication misses mentioned above, it enables *indirect hits*: even if an exact requested view is not available locally or in other servers, the local server can use a different but *correlated* view (either locally stored or fetched from the other servers), so that the remote repository only needs to transmit the pre-encoded differentials between the target view and the correlated view. This leads to a lower overall switching cost than transmission of intra-coded version of the requested view (necessary for replication miss).

Using this new coding structure, we then formulate the content replication problem for multiple movies in order to minimize the switching cost as an *integer linear programming* (ILP) problem and show that it is NP-hard. We then propose a near-optimal replication strategy, which first solves the ILP problem as a relaxed LP problem, and then heuristically rounds the resulting fractional LP solution to integer towards ILP feasibility (using a heuristic called *Minimum Eviction*). Since the LP-rounding algorithm does not scale well to large size problems, we then propose a more computationally efficient solution based on dynamic programming and Lagrangian optimization (DPLO) at the price of only minimal performance penalty. Simulation results show that our replication algorithms reduce switching cost substantially compared to other state-of-the-art content replication schemes.

The paper is organized as follows. We first discuss related work in Section II. We then present the coding structure we design to support static view-switching in Section III. In Section IV, we formulate the content replication problem as an ILP problem and show that it is NP-hard. We present our LP relaxation and rounding algorithm based on Minimum Eviction, and a scalable solution with dynamic programming and Lagrangian optimization in Sections V and VI, respectively. Experimental results are discussed in Section VII. We conclude in Section VIII.

## II. Related Work

### A. Multiview Video Coding for Interactive Applications

There has been much research in multiview video coding (MVC), focusing on compression of all captured frames across time and view, and exploiting both temporal and inter-view correlation to achieve maximal coding gain [6], [7]. However, if the sender transmits all camera-captured views (in a *non-interactive* streaming paradigm where the sender does not consider receiver's feedback when deciding which views to transmit), which depending on system setup can be well over one hundred cameras [8], the transmission bandwidth cost is exorbitant even after employing MVC for compression, since the video bitrate still grows approximately linearly with the number of coded views [6]. This transmission of multiple views looks especially wasteful given that a user typically observes only a *single* view at a time on a conventional 2-D display.

In order to reduce transmission bandwidth, a user in an IMVS system [2] requests and receives from the streaming server only a single coded view for temporal playback, though he can request switches to neighboring views periodically (e.g., every $K$ temporal frames) in an interactive manner; this is a new paradigm that we call *interactive streaming* [9]. Unfortunately, video data pre-encoded using MVC does not provide the data random accessibility required for interactive streaming. In particular, using MVC video for interactive streaming often means that more than one video view must be transmitted just so a single view can be correctly decoded and displayed, resulting in large streaming rates. The reason is that complicated inter-frame dependencies among coded frames across time and view (targeting pure coding gain) reduce the random decodability of the video stream. This point has been well established in the IMVS literature; see [2], [10], [11] for more detailed discussions.

In light of the practical need for new approaches to compress multiview video compactly while maintaining a desired level of data random access, new frame types and frame structures have recently been proposed [2], [12], [13]. A new frame type called SP-frame [12] in the video coding standard H.264 is designed for switching among pre-encoded video streams, thus providing flexibility in decoding. Later, authors in [13] showed that using *distributed source coding* (DSC) principles, one can construct a DSC frame using bit-plane and channel coding, which outperforms the corresponding SP-frame in rate-distortion (RD) performance. The method in [2] then optimizes the construction of a redundant frame structure using I-, redundant P- and DSC frames as building blocks for an IMVS scenario, where a client can interactively switch to an adjacent view every $K$ frames without interrupting temporal video playback, *i.e.* dynamic view-switching. While these works focus their coding design for single server-client communication, we employ here a redundant coding structure to facilitate static view-switching (as opposed to dynamic view-switching in [2]) and to enable indirect hits in a large-scale IMVS network. Further, instead of DSC frame, we employ a more coding-efficient and less complex merge frame for merging of decoding paths [5]. Section VII will show that indirect hit can significantly lower expected transmission rate during an IMVS session.

## B. Large-Scale Video Streaming and Distribution

There have been extensive studies on content replication and replacement strategies for interactive video [14]–[17]. There have also been studies on cooperative caching and server selection in video streaming applications [18]–[20]. However, all these works focus on *single-view* video and hence do not take advantage of *correlation* among views of the same video for saving resources. Therefore, they are not efficient for IMVS. In contrast, we propose a coding structure for multiview video that facilitates static view-switching, and then design content replication strategies to minimize overall switching cost.

More recently, multi-view video streaming has drawn quite some attention in the community [21], [22]. The work in [23] proposes a scheduling algorithm that allows peers to frequently compute the scheduling of multiview segments. The paper [24] studies the problem of achieving low view-switching delay by organizing viewers of different views together. These works essentially treat the multiview video streaming as multiple independent single-view video streams, and have not considered view correlation and switching among views during video playback. Our work extends the field further by considering *both* the coding structure (through novel usage of inter-view P-frames and merge frames to exploit correlation among neighboring views) *and* content replication strategy to achieve nearly minimum expected switching costs.

In our previous work [25], we study multiview video streaming and content replication by proposing a heuristic-based strategy for the delivery of a single multiview video sequence, where we exploit the correlation between the multiple views. In our follow-up work [26], we formulate the optimization problem for multiple videos as an ILP problem. However, the rounding algorithm does not scale to large size problems. In the present work, we extend the coding structure of the multi-view video to include a new encoding scheme for view switches, so that more advanced coding standards, such as H.264, can be supported.

TABLE I
SYMBOLS USED IN PROBLEM FORMULATIONS

| Notation | Meaning |
|---|---|
| $\mathcal{V}$ | Set of servers |
| $E$ | The repository |
| $U$ | Total number of views of movies |
| $S_n^m(i)$ | The $n$th segment for view $i$ of movie $m$ |
| $B_n^m(i)$ | The $n$th chunk for view $i$ of movie $m$ |
| $I_n^m(i)$ | The $n$th I-frame for view $i$ of movie $m$ |
| $P_n^m(i)$ | The $n$th temporal P-frame for view $i$ of movie $m$ |
| $P_n^m(i\|j)$ | The $n$th inter-view P-frame from view $i$ to view $j$ for movie $m$ |
| $W_n^m(i\|j)$ | The $n$th merge frame from view $i$ to view $j$ for movie $m$ |
| $\phi_n^{x,m}(i)$ | The decision variable of replicating chunk $B_n^m(i)$ at server $x$ |
| $\xi_n^m(i,j)$ | Direct hit variable of switching from $B_n^m(i)$ to $B_n^m(j)$ |
| $\zeta_n^m(i,j)$ | Indirect hit variable of switching from $B_n^m(i)$ to $B_n^m(j)$ |
| $\theta_{n,t}^m(i)$ | Temporal switch variable from $B_n^m(i)$ to $B_t^m(i)$ |
| $z_n^m(i)$ | The size of chunk $B_n^m(i)$ |
| $\bar{z}_n^m(i)$ | The size of the first $\Gamma$ frames of segments in $B_n^m(i)$ if the first frame is encoded as I-frame |
| $\bar{z}_n^m(i\|j)$ | The size of the first $\Gamma$ frames of segments in $B_n^m(i)$ if the first frame is encoded as inter-view P-frame (predicted from view $j$) and merge-frame |
| $r$ | The transmission rate between the repository and a local server |
| $\sigma_n^m$ | Average probability that a user is viewing the $n$th chunk, of movie $m$, view $i$ |
| $L(\tau,n)$ | The probability that a user is at chunk $B_n^m(i)$ after $\tau$ temporal switches |
| $\pi^m(i,j)$ | The probability that a user switches from view $i$ to $j$ at the $n$th chunk of movie $m$ |
| $\Pi^m(i)$ | Average probability of view $i$ at the $n$th chunk of movie $m$ |
| $\Omega$ | The probability that a client performs temporal switching |
| $Q^m$ | The probability that movie $m$ is selected for observation |
| $S$ | The expected switching cost |
| $s_n^m(i)$ | The expected cost of temporal switch |
| $u_n^m(i)$ | The expected cost of inter-view switch |
| $C_{n,t}^m(i)$ | The temporal switch cost from chunk $n$ to time $t$ of movie $m$ view $i$ |
| $T_n^m(i,j)$ | The view switch cost from view $i$ to view $j$ of movie $m$ chunk $n$ |
| $D_n^m(i,j)$ | The repository cost of switching from view $i$ to view $j$ of movie $m$ chunk $n$ |

## III. CODING STRUCTURE FOR IMVS

In this section, we introduce the coding structure that we use for pre-encoding multiview video content that is stored in the repository, with subsets of coded frames being distributed among content servers. The structure is designed to facilitate interactive view-switching while the video playback is paused in time (static view switching), so as to enable a look-around visual effect for the viewers. We employ coding tools that have been previously developed to solve the view-switchingproblem [5]. Table I shows a list of important symbols used in this paper.

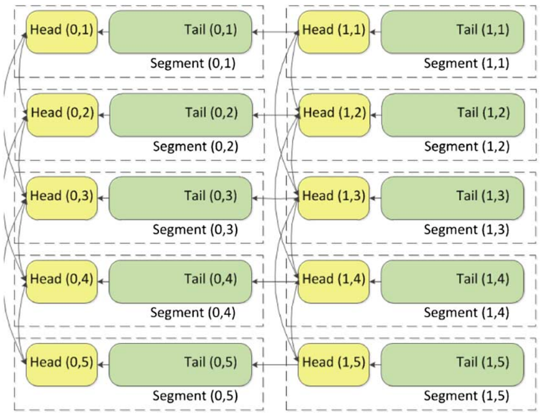


Fig. 2. Dependencies among segments in our proposed multiview video coding structure. Arrows at heads indicate feasible view switches using pre-encoded differentials.

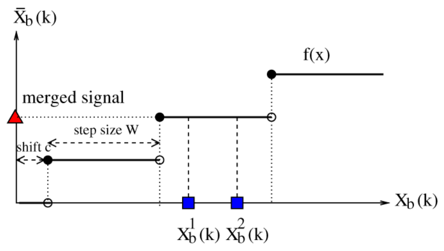


Fig. 3. Piecewise constant function used for signal merging in our proposed merge frame [5].

## A. Frame Structure in Details

To achieve inter-view and temporal switching with good compression efficiency, we propose the following framestructure to pre-encode a given multiview video content. There are a total of $U$ views in the sequence. For any view $i$ ($1 \leq i \leq U$), $K$ consecutive captured video frames in time are encoded into a *segment*. The $n$th segment for view $i$ is labeled as $S_n(i)$, where $0 \leq n \leq N-1$ and $NK$ is the movie length in frames.[2]

In our coding structure, a frame (or picture) at time instant $t$ of view $i$ is labeled as $F_t(i)$, $0 \leq t \leq NK-1$. The segment $S_n(i)$ contains the leading picture $F_{nK}(i)$, called *head* of $S_n(i)$, and the trailing $K-1$ pictures $F_{nK+1}(i)$, $F_{nK+2}(i), \ldots, F_{(n+1)K-1}(i)$, called *tail* of $S_n(i)$.

In the coding structure, $K$ is the *inter-view switching period*, which is the minimum frame interval between two time consecutive static view-switches requested by client. To facilitate view switching, $F_{nK}(i)$ has a *redundant representation*, so that inter-view correlations among nearby viewpoints are exploited during a view-switch. Specifically, for a given *redundant window* $\delta$, the head $F_{nK}(i)$ contains up to $2\delta$ per-encoded differentials using heads $F_{nK}(j)$'s of nearby view $j$'s as predictors, where $j \in \{\max(1, i-\delta), \ldots, \min(U, i+\delta)\}$. Using this coding structure, a view-switch from view $j$ to $i$ only requires transmission of the corresponding pre-encoded differential. We give in Fig. 2 an illustration of a multiview video frame structure for five views $U = 5$ and redundant window $\delta = 2$. Obviously, there is a tradeoff between the total data size and the switching cost by varying the redundant window size $\delta$. We will explore this tradeoff in Section VII.

The head of $S_n(i)$ is represented by multiple compressed versions of the same picture $F_{nK}(i)$:

- One independently coded I-frame $I_{nK}(i)$,
- Multiple differentially coded P-frames $P_{nK}(i)$'s, and
- One *merge* frame $W_{nK}(i)$ [5].

First, a temporal P-frame $P_{nK|nK-1}(i)$ is motion-compensated using a P-frame $P_{nK-1}(i)$ of previous time instant $nK-1$ as predictor. Temporal P-frame $P_{nK|nK-1}(i)$ is for video playback in time in the same view $i$. Then, inter-view P-frames $P_{nK}(i|j)$'s are disparity-compensated, each using I-frame $I_{nK}(j)$ of a nearby view $j$ at the same time instant $nK$ as predictor. Inter-view P-frames $P_{nK}(i|j)$'s are designed for static inter-view switching.

We now describe the key idea in the design of merge frames [5], which are used for effective view switching in our IMVS system. The merge frame $W_{nK}(i)$ is built with multiple P-frames $P_{nK}(i)$'s as predictors using the following procedure. I-frame $I_{nK}(i)$ is the encoding target for $W_{nK}(i)$, which means that a correctly reconstructed merge frame is bit-by-bit equivalent to $I_{nK}(i)$. Each $P_{nK}(i)$ provides *side information* (SI) to help decode $W_{nK}(i)$. Because the reconstructed SI frame $P_{nK}(i)$ and the target $I_{nK}(i)$ are both representations of the same picture $F_{nK}(i)$, the block-wise frequency contents in reconstructed $P_{nK}(i)$ and $I_{nK}(i)$ in Discrete Cosine Transform (DCT) domain are similar. In [5], a *piecewise constant function* (pwc) $f(x)$ is used to merge coefficients of different SI frames to the target. As an example, in Fig. 3 we see two coefficients $X_b^1(k)$ and $X_b^2(k)$ of block $b$ and frequency $k$ for SI frame 1 and 2 respectively, which are close to each other. If the correct *shift parameter* $c$ and *step size* $W$ for a floor function (example of a pwc function) are chosen, then the similar SI coefficients will land on the same step, and the computed value $f(X_b^n(k))$ will be the same (merged) value for all SI frames $n$. [5] discussed the details of how $c$ and $W$ can be chosen for each frequency of each block so that a desired reconstructed value $f(X_b^n(k))$ can be obtained. [5] showed that competitive coding performance can be obtained compared to DSC frames proposed in [13], which involved complex bit-plane and channel coding. SP-frames in H.264 [12] can also be used for the same decoding path merging task. However, the number of secondary SP-frames required at head $F_{nK}(i)$ grows linearly with respect to $\delta$ (while our proposal requires only a single merge frame $W_{nK}(i)$), and each secondary SP-frame is significantly larger than our proposed merge frame due to lossless coding. See [5] for a more detailed discussion.

Thus, by merge frame construction, $I_{nK}(i)$ can be perfectly reconstructed via $W_{nK}(i)$ as long as *one* of the SI frames $P_{nK}(i)$'s is available at decoder. Functionally, $W_{nK}(i)$ serves as a signal merging operator to reconciliate minor differences due to motion/disparity compensation and quantization among P-frames $P_{nK}(i)$'s and $I_{nK}(i)$. This is done so that other frames in turn can simply use the one unified version of $F_{nK}(i)$—reconstructed I-frame $I_{nK}(i)$—as predictor for differential coding without introducing any coding drift. The reconstructed frame is identical no matter which decoding path

[2]We will adopt the convention that superscripts denote movie and/or server indexes, subscripts denote time instants, and brackets denote view number in the sequel.

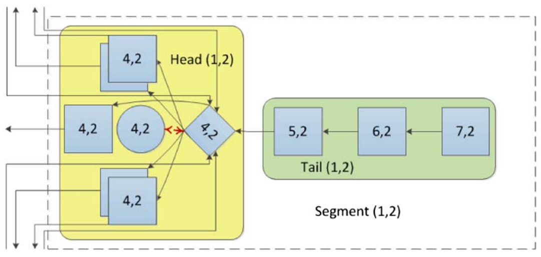


Fig. 4. Implementation of a segment using I-, P-, and merge frames (denoted by circle, squares, and diamonds, respectively).

is actually taken. More specifically, It is mathematically proven in [5] that the decoded quantization bin indices for each DCT block are exactly the same for every decoding path. Hence the resulting decoded frame is bit-by-bit identical, and no coding drift from this point onward.

The merge frame $W_{nK}(i)$ is in practice much smaller than independently coded I-frame $I_{nK}(i)$ [5]. Fig. 4 shows a frame structure for segment $(1,2)$ with the different types of frames discussed above.

### B. *Benefits of Coding Structure in Interactive Systems*

We now discuss how the redundant frame structure described above is used for a large-scale distributed IMVS network. A server in the network may choose to replicate a segment, which will include the tail of the segment, plus the I-frame of the head of the segment. All prediction differentials (inter-view P-frames and merge frames) of the head of the segment are only stored in the repository. Generally, the viewer tries to get the data from local servers, and accesses the repository only if this is not possible. If a viewer requests a view $j$ after observing view $i$, where $|i-j| \leq \delta$, only the inter-view P-frame $P_{nK}(j|i)$ and the merge frame $W_{nK}(j)$ are needed from the repository for the decoder to switch views, if the target I-frame $I_{nK}(j)$ is not available locally. On the other hand, if $|i-j| > \delta$, then the much larger independently coded I-frame $I_{nK}(j)$ is needed.

The repository transmission is generally much slower than local server transmission. Therefore, the goal of minimizing switching cost is equivalent to minimizing required repository transmission. Thus, to avoid repository transmission of $I_{nK}(j)$ during an inter-view switch, we do the following. If a neighboring server has replicated I-frame $I_{nK}(l)$, where $|l-j| \leq \delta$, then the server first sends $I_{nK}(l)$ to the local server (with negligible delay), while the repository transmits the smaller P-frame $P_{nK}(j|l)$ and merge frame $W_{nK}(j)$ to the local server. This is called *indirect hit*: a local server which does not have the requested view $j$ but has a correlated view $l$, can reduce the switching cost of repository transmitting $I_{nK}(j)$ to the cost of repository transmitting $P_{nK}(j|l)$ and $W_{nK}(j)$.

To summarize, there are four possible transmission cases during a static inter-view switch from $i$ to $j$, depending on the content replicated in servers. In order of increasing costs, they are:

1) *Direct hit*: When a neighboring server or the local server has the exact segment $S_{nK}(j)$, the replicated I-frame $I_{nK}(j)$ can be forwarded locally. This is of negligible cost.
2) *Differential transmission*: The repository transmits pre-encoded differentially coded P-frame $P_{nK}(j|i)$ and the merge frame $W_{nK}(j)$.
3) *Indirect hit*: A neighboring server $y$ has replicated a *correlated* frame $I_{nK}(l)$, which is forwarded to the local server, and the repository transmits only P-frame $P_{nK}(j|l)$ and merge frame $W_{nK}(j)$ to the local server.
4) *Replication miss*: No server has exact or correlated frames, and the repository transmits independently-coded I-frame $I_{nK}(j)$ to the local server.

We illustrate the four transmission cases in Fig. 5. Fig. 5(a) shows the flow diagram of an inter-view switch. In the example of Fig. 5(b), We consider a multiview video with 7 independent views. View 4 and view 7 are replicated by the content servers, while the rest of the views are stored only in the repository. We consider the redundant window $\delta = 1$ in this example, and the user is currently watching view 4. Switching to view 6 is a direct hit because view 6 is replicated by the content servers. Switching to view 3 is a differential transmission because view 3 is within the redundant window of view 4, and only the merge frame and differential P-frame need to be transmitted. Switching to view 7 is an indirect hit because view 7 is within the redundant window of another replicated view, view 6. Therefore view 6 is delivered from a content server, and the merge frame and differential P-frame are delivered from the repository. Switching to view 1 is a replicate miss because no server replicates the exact or correlated views. In this case the repository transmits the complete view to the user. Fig. 5(a) shows the flow diagram of an inter-view switch.

Due to indirect hit and differential transmission with our coding structure, the switching cost from the repository can be substantially reduced. Note finally that because there are no differentially coded P-frames $P_{tK|nK}(i)$, $n \neq t$, there are only two possible costs for temporal switching, direct hit or replication miss, similar to conventional content replication methods in interactive video applications.

## IV. Problem Formulation and Complexity

In this section, we formulate the distributed content replication problem for a minimal cost IMVS system as an ILP problem. We first define the decision variables, followed by the constraints and optimization objective. We finally prove that the problem is NP-hard.

### A. *Decision Variables*

We consider the distributed content replication problem with $M$ movies of $U$ captured views each. Let $\mathcal{V}$ and $E$ be the set of servers and the repository, respectively. For efficient replication of movie $m$, sets of $H$ consecutive segments in time are grouped as "*chunks*", which is the basic replication unit. More precisely, the chunk $B_n^m(i) = \{S_{nH}^m(i), S_{nH+1}^m(i), \ldots, S_{(n+1)H-1}^m(i)\}$ includes the segments $S_\eta^m(i)$'s, where $nH \leq \eta \leq (n+1)H - 1$. When a server replicates $B_n^m(i)$, for each segment $S_\eta^m(i)$ in $B_n^m(i)$ it replicates I-frame $I_{\eta K}^m(i)$ and temporal P-frames $P_{\eta K+1|\eta K}^m(i), \ldots, P_{(\eta+1)K-1|(\eta+1)K-2}^m(i)$ in its local storage. Inter-view P-frames $P_{\eta K}^m(i|j)$'s and merge frame $W_{\eta K}^m(i)$ reside only at the repository.

Each server must decide which chunk(s) of movie $m$ to replicate given its own (limited) storage size. Let $\phi_n^{x,m}(i) \in \{0,1\}$

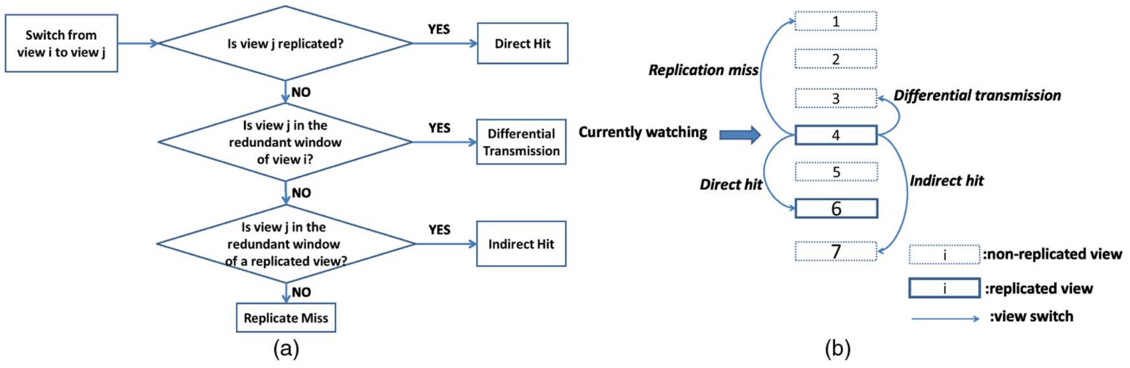


Fig. 5. Four transmissions in inter-view switches. (a) Flow diagram. (b) An example.

be a binary variable indicating whether to replicate chunk $B_n^m(i)$ of movie $m$ at server $x$. When watching a segment in chunk $B_n^m(i)$ of movie $m$, a user can request an inter-view or temporal switch, and IMVS system has to decide where (content server or repository) to get the requested multiview data from. For inter-view switching from view $i$ to $j$, we first define $\xi_n^m(i,j) \in \{0,1\}$ to be the binary variable indicating whether to directly pull the requested view $j$ from a local server (direct hit). If chunk $B_n^m(j)$ is not replicated locally, we define $\zeta_n^m(i,j) \in \{0,1\}$ to be the binary variable indicating whether to pull a correlated intermediate view $l$ from a local server and pull the pre-coded differential between the intermediate view $l$ and the requested view $j$ from the repository (indirect hit). Clearly, $\xi_n^m(i,j) = \zeta_n^m(i,j) = 0$ means that requested view $j$ must be pulled entirely from the repository (differential transmission or replication miss, depending on view distance $j$ from $i$).

For temporal switching (i.e., switching to a temporal segment of the same view *not* in the current chunk $B_n^m(i)$), we define $\theta_{n,t}^m(i) \in \{0,1\}$ as the binary variable indicating whether to pull the requested multiview data from a local server for a temporal switch from chunk $B_n^m(i)$ to $B_t^m(i)$.

## B. Linear Constraints

We discuss now the system constraints in our content replication problem. For movie $m$, first let $z_n^m(i)$ be the size of chunk $B_n^m(i)$. Further, let the storage capacity of server $x$ be denoted $c^x$. Using the decision variables introduced above, we can write the following capacity constraint for server $x$:

$$\sum_m \sum_n \sum_i \phi_n^{x,m}(i) z_n^m(i) \le c^x, \quad \forall x, \tag{1}$$

which means that the sum of replicated content cannot exceed the storage capacity.

Then we observe that the temporal switch variable $\theta_{n,t}^m(i)$ can be 1 only if there is at least one server $x$ replicating chunk $B_t^m(i)$. Thus, we can write:

$$\theta_{n,t}^m(i) \le \sum_x \phi_t^{x,m}(i), \quad \forall m,n,t,i. \tag{2}$$

Similarly, the direct inter-view switch variable $\xi_n^m(i,j)$ can be 1 only if there is a server $x$ replicating chunk $B_n^m(j)$, i.e.,

$$\xi_n^m(i,j) \le \sum_x \phi_n^{x,m}(j), \quad \forall m,n,i,j. \tag{3}$$

Then, the indirect inter-view switch variable $\zeta_n^m(i,j)$ can be 1 only if there is a server $x$ replicating chunk $B_n^m(l)$, where view $l$ is in the window $j-\delta \le l \le j+\delta$, so that, the repository only needs to send the inter-view P-frame $P_\eta^m(j|l)$ and the merge frame $W_\eta^m(j)$. Thus, we can write

$$\zeta_n^m(i,j) \le \sum_x \sum_{l=j-\delta, l\neq j}^{l=j+\delta} \phi_n^{x,m}(l), \quad \forall m,n,i,j. \tag{4}$$

Finally, we want to ensure that indirect and direct hits are not selected simultaneously. Thus, for a given inter-view switch from $i$ to $j$, we write:

$$\xi_n^m(i,j) + \zeta_n^m(i,j) \le 1. \tag{5}$$

We can see that all the above system constraints are linear with respect to the decision variables.

## C. Inter-view & Temporal Switch Model

Before defining the objective function of our optimization problem, we describe the probabilistic model that we use to model the likelihood that a user chooses different inter-view and temporal switches. For simplicity, we assume users select inter-view switches independent of time and temporal switches independent of view, and hence we model them separately.

We start with the temporal switch model. For movie $m$, let $p_{n,t}^m$ be the probability that a user *temporally* switches from $n$-th chunk $B_n^m(i)$ to $t$-th chunk $B_t^m(i)$ for *any* view $i$. We assume that the users start an IMVS session at first chunk $B_0^m(i)$ of some view $i$. At an arbitrary time after video playback has started, let $\sigma_n^m$ be the average probability that a user is observing chunk $n$ of movie $m$. We derive $\sigma_n^m$ in terms of the switching probability $p_{n,t}^m$ as follows.

Let $T$ be a discrete random variable denoting the total number of temporal switches that a user have already made at the current time instance (including both sequential temporal playback and jumps). Denote the probability mass function of $T = \tau$ as $f_T(\tau)$. Let $L^m(\tau,n)$ be the probability that a user is at the $n$-th chunk after $\tau$ temporal switches, given that the user starts from the first chunk. Considering the transitions from chunks to chunks, $L^m(\tau,n)$ can be derived recursively as

$$L^m(\tau,n) = \begin{cases} 1, & \text{if } \tau = n = 0; \\ \sum_k L(\tau-1,k) p_{k,n}^m, & \text{otherwise.} \end{cases} \tag{6}$$

The average probability $\sigma_n^m$ can then be expressed as[3]

$$\sigma_n^m = \sum_\tau L^m(\tau, n) f_T(\tau). \tag{7}$$

Clearly, $\sigma_n^m$ is a probability measure because it satisfies $\sum_n \sigma_n^m = 1$.

For inter-view switches, we let $\pi^m(i,j)$ be the probability that a user switches from view $i$ to $j$ independent of time. Inter-view switch probabilities $\pi^m(i,j)$'s thus constitute an irreducible, aperiodic and positive-recurrent Markov chain, and we can compute the average state probability $\Pi^m(i)$ of view $i$ by: i) performing eigen-decomposition of the state transition matrix, and ii) identifying the eigenvector associated with eigenvalue 1.

Finally, we let $\Omega$ be the probability that in case of a switch, a client performs temporal switching as opposed to inter-view switching, independent of time and view; i.e., the viewer switches to a different view with probability $1-\Omega$.

### D. Objective Function

For the movie $m$, let $s_n^m(i)$ and $u_n^m(i)$ be the cost of temporal and inter-view switching, respectively, stemming from the chunk $B_n^m(i)$. Let $Q^m$ be the probability that the movie $m$ is selected for playback. The expected switching cost $S$ can then be expressed as

$$S = \sum_m \sum_i \sum_n Q^m \sigma_n^m \Pi^m(i) \left[\Omega s_n^m(i) + (1-\Omega) u_n^m(i)\right]. \tag{8}$$

Our objective is to minimize the expected switching cost $S$ by deciding: i) which server to replicate each chunk $B_n^m(i)$ ($\phi$'s), and ii) where to pull content from upon a temporal switch ($\theta$'s) or an inter-view switch ($\xi$'s and $\zeta$'s). The problem is equivalent to computing the following:

$$\min_{\{\phi\},\{\theta\},\{\zeta\},\{\xi\}} S. \tag{9}$$

The temporal switch cost $s_n^m(i)$ in $S$ can be expressed as the sum of all temporal switch costs $C_{n,t}^m(i)$'s to some chunk index $t$:

$$s_n^m(i) = \sum_t p_{n,t}^m C_{n,t}^m(i). \tag{10}$$

Let the transmission rate between the repository and a local server be denoted as $r$. Let $\bar{z}_t^m(i)$ be the size of first $\Gamma$ frames of the segments in chunk $B_t^m(i)$, *if* the first frame is encoded as I-frame, for video client rebuffering[4]. Each temporal switch cost $C_{n,t}^m(i)$'s can then be written simply in terms of temporal switch variable $\theta_{n,t}^m(i)$:

$$C_{n,t}^m(i) = \theta_{n,t}^m(i)\epsilon + (1-\theta_{n,t}^m(i))\bar{z}_t^m(i)/r. \tag{11}$$

[3]In practice, in the summation we count only terms where $f_T(\tau)$ is non-negligible, so that the number of terms in the summation is finite.

[4]$\Gamma$ can be set according to the default client video player behavior. For example, $\Gamma = 1$ means the player will play back video as soon as there is a complete frame in the buffer.

Equation (11) says that the temporal switch cost is equal to some small $\epsilon$ if the chunk $B_t^m(i)$ is replicated in a neighboring server, and it is equivalent to the amount of time required to transmit the first $\Gamma$ frames of the segment, $\bar{z}_t^m(i)/r$, to the local server if the new segment must be pulled from the repository.

Then, we can write inter-view switch cost $u_n^m(i)$ as the sum of all view-to-view costs $T_n^m(i,j)$ to some view $j$:

$$u_n^m(i) = \sum_j \pi^m(i,j) T_n^m(i,j). \tag{12}$$

The view-to-view cost $T_n^m(i,j)$ has a small value $\epsilon$ if there is a direct hit in the local server ($\xi_n^m(i,j) = 1$). If there is an indirect hit ($\zeta_n^m(i,j) = 1$), then the repository must transmit a P-frame and a merge frame, resulting in cost no larger than $\epsilon + \bar{z}_n^m(j|j \pm \delta)/r$; we bound the size of the first $\Gamma$ frames—starting with a inter-view P-frame $P_\eta^m(j|l)$ and merge frame $W_\eta^m(j)$—for any intermediate view $l$, $j-\delta \le l \le j+\delta$, with $\bar{z}_n^m(j|j\pm\delta)$.

Finally, if repository must transmit everything ($\xi_n^m(i,j) = \zeta_n^m(i,j) = 0$), then repository cost $D_n^m(i,j)$ depends on whether the target view $j$ is within the prediction window $[i-\delta, i+\delta]$ (differential transmission) or not (replication miss):

$$D_n^m(i,j) = \begin{cases} \bar{z}_n^m(j|i)/r, & \text{if } |i-j| \le \delta; \\ \bar{z}_n^m(j)/r, & \text{otherwise}. \end{cases} \tag{13}$$

Summarizing the above, the view-to-view cost is written as

$$\begin{aligned} T_n^m(i,j) = {} & \xi_n^m(i,j)\epsilon + \zeta_n^m(i,j)(\epsilon + \bar{z}_n^m(j|j-\delta)/r) \\ & + \left[1 - \xi_n^m(i,j) - \zeta_n^m(i,j)\right] D_n^m(i,j). \end{aligned} \tag{14}$$

Since $\epsilon$, $\bar{z}_n^m(j|j-\delta)$ and $D_n^m(i,j)$ are fixed, it is clear that the temporal switch cost $C_{n,t}^m(i)$ and the inter-view switch cost $T_n^m(i,j)$ are linear in the decision variables. Thus, the expected temporal and inter-view switch cost $s_n^m(i)$ and $u_n^m(i)$ are also linear, and the objective function $S$ is also linear. Since all the constraints are also linear, our problem is an Integer Linear Programming problem (ILP).

### E. NP-Hardness Proof

Unfortunately, our ILP optimization problem is NP-hard. We prove that by showing a special case of the problem can be mapped to the known NP-complete problem *bin packing* [27], which is the following. Given a bin capacity $W$, a list of items of sizes $a_1, \ldots, a_Z$, and integer $B$, is there a capacity-preserving item-to-bin assignment so that $B$ or fewer bins are required?

Consider a special case of our problem where there are $B$ servers in the IMVS network, each of storage size $W$, and where the multiview video has only one view. Thus, the client can only perform temporal switching. Suppose each chunk $B_n$ of $Z$ chunks has size $a_z$, and each chunk is requested with equal likelihood. If there is a content replication strategy to fit all chunk in the servers (reflected in the resulting cost $S$ since no repository transmission is required), then there is a capacity-preserving item-to-bin assignment to fit all items in $B$ or fewer bins. It corresponds to solving a NP-Complete binary decision bin packing problem. Finally, our optimization problem is a general version of the previous problem; thus it is at least as hard as the NP-complete binary decision bin packing problem, and hence it is NP-hard.

## V. LP Relaxation and Rounding with Minimum Eviction

In this section, we present a first algorithm termed *Minimum Eviction*[5] which provides an approximate solution to the formulated ILP problem in Section IV. We first discuss the principles of relaxing the ILP problem to an LP one. Given the solution of the LP problem, we next discuss Minimum Eviction to round the fractional LP solution to integers for a feasible approximate solution to the original ILP problem.

### A. Principles of LP Relaxation

Though the ILP problem posed earlier has linear constraints and objective, it is difficult to solve because of the integer constraints. If we remove these integer constraints, we can solve the resulting LP problem using one of several known algorithms (like Simplex) in polynomial time [27]. The resulting objective function value $S^*$ is called a *super-optimal* solution; i.e., $S^* \leq S^o$, where $S^o$ is the true optimal solution value to the original ILP problem. The reason is that LP problem is a relaxed version of the original ILP problem of Equation (9) with fewer constraints.

If we perform rounding to the LP solution so that the integer constraints are satisfied, we have a (likely sub-optimal) solution that is feasible with objective value $S^a$. The approximation error $e$ from the true optimal solution is bounded as follows:

$$e = |S^a - S^o| \leq |S^a - S^*| \tag{15}$$

The proof is straight-forward:

$$S^o \leq S^a$$
$$S^o - S^* \leq S^a - S^*$$

Then (15) follows because $S^* \leq S^o$.

Thus, the LP solution provides us with an *a posteriori* approximation bound to quantify the quality of our rounded solution. We discuss next how the LP solution also provides additional information so that we can perform integer rounding to a good approximate solution.

### B. Rounding Heuristic: Minimum Eviction

Given an LP solution, we can classify the storage variables $\phi_n^{x,m}(i)$'s into two classes: 1) *Primary variables*, which are the fractions $\phi_n^{x,m}(i)$'s of the same chunk $B_n^m(i)$ that sum to one across servers, i.e., $\sum_x \phi_n^{x,m}(i) = 1$; and 2) *Secondary variables*, which are the fractions $\phi_n^{x,m}(i)$'s of the same chunk $B_n^m(i)$ when $\sum_x \phi_n^{x,m}(i) < 1$ instead. The LP solution tells us that the primary variables are more important than the secondary ones, because the chunks are stored in entirety in servers. The heuristic *Minimum Eviction* algorithm essentially tries to fit as many primary variables in server storage as possible by iteratively considering fractional chunks, starting with the one of largest size first, as follows:

[5]"Eviction" is a common term used in caching literature to mean removal of less useful contents for storage of more useful ones when the cache capacity is full.

1) Identify the storage variables $\phi_n^{x,m}(i)$'s that are equal to 1. These are *stable* assignments and will not be changed further.
2) Find the target fractional primary variable $\phi_n^{x,m}(i) < 1$ representing the largest fractional chunk. Round this fraction up, and round the corresponding variables $\phi_n^{y,m}(i)$'s in other servers $y$ down.
3) If rounding up in step 2 results in a storage constraint violation for server $x$, evict secondary variables in the order of decreasing fractional chunk sizes until: i) the constraint is met, or ii) no more secondary variables are left.
4) If the storage constraint in server $x$ is still violated, evict the unstable primary variables $\phi_n^{x,m}(i) < 1$ in server $x$ in the order of decreasing fractional chunk sizes until: i) the constraint is met, or ii) no more unstable primary variables are left.
5) If the storage constraint in server $x$ is still violated, evict the target unstable variable $\phi_n^{x,m}(i)$ instead.
6) If there are no more primary variables, then round down all remaining fractional variables and the algorithm finishes. Otherwise, go back to Step 1.

The key idea is that, by attempting to round up storage variable with the largest fractional chunk size, it is either kept in server $x$ or removed from the servers, but it is never moved from one server to another.

## VI. DPLO: Dynamic Programming Followed by Lagrangian Optimization

In the previous section, we propose a solution based on LP relaxation and rounding. Because the LP-rounding algorithm does not scale well to large number of variables, we propose here a more scalable solution based on dynamic programming and Lagrangian optimization (DPLO), which suffers little loss in performance. DPLO has two stages. In the first stage, it uses DP to calculate the maximum replication benefit of each movie chunk in a server, subject to the server storage constraint. In the second stage, it uses Lagrangian optimization based on the replication benefits to optimally store the chunks in each server.

### A. A Benefit Measure for Chunk Replication

We solve the ILP problem formulated in the previous section in the following manner. Let $\phi_n^{x,m}(i) \in \{0,1\}$ be the binary decision variable indicating whether server $x$ replicates the chunk $B_n^m(i)$. Since there is no additional benefit of replicating a chunk more than once among servers (due to negligible cost between servers), we have

$$\sum_x \phi_n^{x,m}(i) \leq 1, \quad \forall m,n,i. \tag{16}$$

In other words, the chunk $B_n^m(i)$ will be replicated at most once among servers.

We analyze the cost and benefit of replicating the chunk $B_n^m(i)$ as follows. Replicating the chunk $B_n^m(i)$ consumes $z_n^m(i)$ server storage space. The benefit $G_n^m(i)$ of replicating chunk $B_n^m(i)$ among servers, when the replicated chunk of the nearest view $\rho$ is $B_n^m(\rho)$, can be written as

$$\begin{aligned} &G_n^m(i,\rho) \\ &= Q^m \Bigg\{ \Omega \sum_t \sigma_t^m(i) p_{t,n}^m(i) (\bar{z}_n^m(i)/r - \epsilon) \\ &\quad + (1-\Omega) \Bigg[ \sum_j \Pi_n^m(j) \Big( \pi_n^m(j,i) g_n^m(j,i,\rho) \\ &\quad + \sum_{k \mathrm{s.t.} k \neq i, |k-i| \leq \delta} \pi_n^m(j,k) h_n^m(j,k,i,\rho) \Big) \Bigg] \Bigg\}. \end{aligned} \tag{17}$$

In words, the equation states that the benefit $G_n^m(i,\rho)$ of replicating chunk $B_n^m(i)$ is the difference in the cost between replication miss (i.e., $\bar{z}_n^m(i)/r$) and direct hit (i.e., $\epsilon$) during a temporal switch from $t$ to $n$, *plus* the difference in transmission cost during an inter-view switch. For inter-view switch, we divide the potential benefit into two cases: i) view-switch from view $j$ to $i$, and ii) view-switch from view $j$ to some other view $k$, $k \neq i$, and view $i$ is used as an intermediate view during an indirect hit. For the first case, we write the benefit as $g_n^m(j,i,\rho)$:

$$g_n^m(j,i,\rho) = \begin{cases} \bar{z}_n^m(i|i-\delta)/r, & \text{if } |j-i| \leq \delta \text{or} |\rho - i| \leq \delta; \\ \bar{z}_n^m(i)/r - \epsilon, & \text{otherwise.} \end{cases} \tag{18}$$

The equation above states that if the view-switch from $j$ to target $i$ is within $\delta$, or if the view of an already replicated chunk $B_n^m(\rho)$ is also within $\delta$ of target $i$, then the benefit is only the difference between a transmission cost of an indirect hit $\bar{z}_n^m(i|i-\delta)/r + \epsilon$ and a direct hit $\epsilon$, i.e., $\bar{z}_n^m(i|i-\delta)/r$. Otherwise, the benefit is the difference between the cost of a replication miss $\bar{z}_n^m(i)/r$ and a direct hit $\epsilon$.

For the second case that represents the switching from view $j$ to a target view $k$, where $k \neq i$, we write the potential benefit of using view $i$ in an indirect hit as $h_n^m(j,k,i,\rho)$:

$$h_n^m(j,k,i,\rho) = \begin{cases} 0, \text{if } |\rho - k| \leq \delta \text{ or } |j-k| \leq \delta; \\ \bar{z}_n^m(i)/r - \bar{z}_n^m(k|k-\delta)/r - \epsilon, \text{otherwise.} \end{cases} \tag{19}$$

The above states that if the view of an already replicated chunk $B_n^m(\rho)$ is within $\delta$ of target $k$, or if the current view $j$ is within $\delta$ of target $k$, then the cost of view-switching from $j$ to $k$ is already no worse than an indirect hit[6], and replicating chunk $B_n^m(i)$ brings no additional benefit. Otherwise, the benefit is the difference between the cost of a replication miss $\bar{z}_n^m(i)/r$ and the cost of an indirect hit $\bar{z}_n^m(k|k-\delta)/r + \epsilon$.

Given the above cost/benefit analysis for each chunk, we can derive an algorithm that operates in two stages as follows. In the first stage, we determine how the storage space should be optimally distributed among different views of chunks of the same movie $m$ and time index $n$. In the second stage, we determine how the storage space should be distributed among chunks of different movies and different time indices. We will demonstrate in Section VII that this "divide-and-conquer" strategy is computationally much more efficient than the *Minimum Eviction* algorithm discussed in Section V.

[6]If $k = \rho$, then it is a direct hit with cost $\epsilon$, which is smaller than an indirect hit $\bar{z}_n^m(k|k-\delta)/r - \epsilon$.

## B. Stage 1: Dynamic Programming to Calculate Replication Benefits of Movie Chunks

We now discuss the first stage, namely a dynamic programming (DP) algorithm to find the optimal selection of chunks of the same movie $m$ and time index $n$, given the available server storage space. First, let $K_n^m(c_n^{1,m},\ldots,c_n^{X,m})$ be the maximum possible benefit achieved by the IMVS network by selecting which views of movie $m$, chunk $n$ to replicate under the constraints of available storage capacities $c^1,\ldots,c^X$ in $X$ servers. We can solve $K_n^m(c^1,\ldots,c^X)$ using recursive function $\kappa_n^m(i,k,c^1,\ldots,c^X)$, which is defined to be the maximum benefit given optimal replication decision has been made for view $< i$ and the most recent replicated view is $k$:

$$K_n^m(c^1,\ldots,c^X) = \kappa_n^m(1,-\delta,c^1,\ldots,c^X). \tag{20}$$

The variable $\kappa_n^m(i,k,c^1,\ldots,c^X)$ can be defined recursively as follows:

$$\begin{aligned} &\kappa_n^m(i,k,c^1,\ldots,c^X) = \max \Big( \kappa_n^m(i+1,k,c^1,\ldots,c^X), \\ &\max_{1 \leq x \leq X, c^x \geq S_n^m(i)} \{ G_n^m(i,k) + 1(i<U) \kappa_n^m \\ &\quad \times (i+1,i,c^1,\ldots,c^x - S_n^m(i)r,\ldots,c^X) \} \Big). \end{aligned} \tag{21}$$

In words, the above equation says that $\kappa_n^m(i,k,c^1,\ldots,c^X)$ is the larger of $\kappa_n^m(i+1,k,c^1,\ldots,c^X)$ if chunk $B_n^m(i)$ has not been replicated, and benefit $G_n^m(i,k)$ of chunk $B_n^m(i)$ *plus* future recursive cost, if $B_n^m(i)$ has been replicated. Note that the recursive term has a smaller remaining storage size $c^x - S_n^m(i)r$ for server $x$, and that the most recent replicated view has been updated to $i$.

We solve $K_n^m(c^1,\ldots,c^X)$ with arguments $c^1,\ldots,c^X$, where $c^x$ is the total storage capacity of server $x$. Equation (21) can be solved using DP, where solution to $\kappa_n^m(i,k,c^1,\ldots,c^X)$ is stored in entry $[i][k][c^1]\ldots[c^X]$ of a DP table, so that a future call to $\kappa_n^m()$ with the same argument can be simply looked up. The time complexity of the algorithm is then the number of steps that are necessary to compute each DP table entry ($O(X)$ using Equation (21)), times the number of entries in the DP table (which is $N^2 \prod_{x=1}^X c^x$).

Nonetheless, the size of the DP table can be large, leading to large computation costs. We can reduce the complexity by a *rounding factor $R$* as follows. First, the storage sizes for the arguments of the first call $K_n^m(c^1,\ldots,c^X)$ are each scaled and rounded *down* by $R$, i.e., $\lfloor c^1/R \rfloor,\ldots,\lfloor c^X/R \rfloor$. Then, in Equation (21), when the chunk $B_n^m(i)$ is replicated in server $x$, the reduction in size $S_n^m(i)$ is then scaled and rounded *up* by $R$, i.e., $\lceil S_n^m(i)/R \rceil$. In doing so, an entry $[i][k][\varsigma^1]\ldots[\varsigma^X]$ in the DP table now represents the solution if storage sizes of *at least* $\varsigma^1 * R,\ldots,\varsigma^X * R$ are available for chunks $B_n^m(i)$'s of view $i$, each chunk being of size *no larger than* $\lceil S_n^m(i)/R \rceil R$. The rounding directions are chosen so that the obtained solution remains feasible in the original problem without rounding. Large rounding errors due to large $R$, however, would mean that more storage space in servers are left unused, leading to larger approximation error in the obtained solution. The benefit on the other side is a reduction in DP table size[7] by factor $R^X$, thus reducing the overall complexity of the algorithm.

We note the following two observations about the DP algorithm. First, the computation for $K_n^m(c^1,\dots,c^X)$ can be carried out independently for chunks of the different movies in the stage 1 of our algorithm. And at different time instants, leading to an efficient parallel implementation. Second, the DP tables hosting the computed results in stage 1 are only constructed *once* (for a given rounding factor $R$). In the Lagrangian optimization in stage 2 of the optimization algorithm, the *same* DP tables can be reused without re-computation, although the optimization has to be performed multiple times in search for the appropriate Lagrange multipliers $\lambda^x$'s,.

## C. Stage 2: Lagrangian Optimization for Chunk Storage

We have described above how $K_n^m(c_n^{1,m},\dots,c_n^{X,m})$ can be solved with a DP algorithm. The overall constrained optimization problem can then be written as

$$\max_{\{c_n^{x,m}\}} \sum_m \sum_n K_n^m(c_n^{1,m},\dots,c_n^{X,m})$$
$$\text{s.t.} \quad \sum_m \sum_n c_n^{x,m} \le c^x, \quad \forall x. \tag{22}$$

Instead of solving Equation (22) directly, we propose to solve its Lagrangian version using $X$ Lagrange multipliers $\lambda^x > 0$:

$$\max_{\{c_n^{x,m}\}} \sum_m \sum_n \left[ K_n^m(c_n^{1,m},\dots,c_n^{X,m}) - \sum_x \lambda^x c_n^{x,m} \right]. \tag{23}$$

We can see clearly that Equation (23) is *separable*, i.e., for fixed $\lambda^x$'s we can solve for optimal $c_n^{x,m}$'s for chunks $B_n^m(i)$ of all view $i$'s independently of other chunks, without loss of optimality. In other words, we can solve independently the following set of problems:

$$\max_{\{c_n^{x,m}\}} \left[ K_n^m(c_n^{1,m},\dots,c_n^{X,m}) - \sum_x \lambda^x c_n^{x,m} \right], \quad \forall m,n. \tag{24}$$

This means that Equation (24) for different movies and at different time instants can also be solved independently in a parallel implementation, similarly to the DP algorithm in the stage 1 of the optimization algorithm. Equation (24) for a given pair of movie index $m$ and time instant $n$ can be solved easily. The only remaining task is to find $\lambda^x$ so that the operational storage sizes are as close to the original constraints $c^x$'s, without exceeding them. This can be done, for example, using binary search on the real positive number line.

In summary, our optimization procedure of DPLO is as follows:

1) Using $c^1,\dots,c^X$ as argument, construct functions $K_n^m()$ for all $m$ and $n$ with Equation (20). Use $R$ to control complexity.
2) Initialize $\lambda^x > 0, 1 \le x \le X$.
3) Solve Equation (23) for $\{c_n^{x,m}\}$ given $\lambda^x$'s. Increase $\lambda^x$ if $\sum_m \sum_n c_n^{x,m} > c^x$. Otherwise, decrease $\lambda^x$ only if $c^x - \sum_m \sum_n c_n^{x,m} > \delta$.
4) Repeat Step 3 if a $\lambda^x$ has been updated.

[7]We note that integer rounding to reduce DP table size is a standard technique in *polynomial-time approximation scheme* (PTAS) in combinatorial optimization [27].

## VII. Illustrative Simulation Results

In this section, we present simulation results to demonstrate the performance of our multiview coding and content replication strategies.

### A. Simulation Environment, Comparison Schemes and Performance Metrics

In our simulation, we first run a set of experiments on the MPEG multiview videos to estimate the respective sizes of I-frames, temporal P-frames, inter-view P-frames, and merge frames. In the experiments, we use the multiview video sequences of *Kendo*, *Champagne tower* and *Pantomime*, provided by the Tanimoto Laboratory, Nagoya University [28]. Views are coded into our proposed frame structure using a H.264 codec: I- and P-frames are coded using conventional H.264 tools, while merge frames are coded using the methodology described in [13]. The quantization parameter is fixed at 40 for all frames for constant visual quality. Note that the benefits offered by our optimized replication algorithm are not dependent on the actual video coding algorithms deployed. The performance gain may vary depending on the version of the video coding standards used in the experiments, but the results would be qualitatively the same. We normalize the size of each I-frame, merge frame, temporal and inter-view P-frame into block units. Then we randomly generate frames for the movies with the sizes and distributions according to the experiment results. The size of I-frames is distributed with mean 4 units. The size of a temporal P-frame is distributed with mean 1 units. The size of a merge frame plus a inter-view P-frames between view $i$ and view $j$ equals to $2 + 0.15|i - j|$ units. Unless otherwise stated, we use the baseline parameters as shown in Table II to represent the system settings and the different costs in the IMVS system. The popularity of the movies follows the Zipf distribution with parameter $s$ (i.e., the access probability is proportional to $1/i^s$, where $i$ is the movie index). We have also run our simulations using different popularity distributions. The results of those simulations are qualitatively the same as those based on the Zipf distribution, and hence they are not shown for the sake of brevity.

We compare our replication strategies *Minimum Eviction* and *DPLO* with the following schemes:

- Local Greedy [29]: Local greedy is a state-of-the-art replication strategy, in which each server replicates chunks with high utility. In other words, popular movies and views are most likely to be replicated. A few servers replicate medium popular movies and views, and unpopular ones are only stored at the repository. In our implementation, we use half of the storage in each server to replicate the most popular chunks, and the other half the storage in each server to replicate the medium popular ones. The unpopular ones are not replicated by the servers.

TABLE II
BASELINE PARAMETERS IN OUR SIMULATION

| Parameter | Baseline value |
|---|---|
| Number of movies | 70 |
| Number of views | 10 |
| Number of chunks per movie | 6 |
| Average chunk size | 300 units |
| Cost of replicate miss | 350 |
| Cost of indirect hit | 100 |
| Cost of differential transmission | 70 |
| Number of servers | 30 |
| Server storage size | 12000 units |
| View switch tendency | 0.5 |
| Rounding parameter $R$ | 2 |
| Zipf parameter $s$ | 0.5 |

- Random: Random replication is a simple replication method. Each server randomly replicates chunk of different views and different movies up to using fully the available storage.

We evaluate the performance of our proposed algorithms and compare them with other schemes in terms of run time, switching cost and request performance.

- Run time: The run time is defined as the total number of seconds needed to compute the replicating strategy. We conduct our simulations on a 64-bit desktop computer with Intel Core i7-2600 CPU@3.40 GHz and 8 GB RAM running Windows 7 operating system. We set the normalized run time unit to be 10 seconds in this experiment. We are particularly interested in the normalized running time of *DPLO* compared to *Minimum Eviction*.
- Switching cost: Then, the switching cost is our objective metric in the problem formulation. We study the switching cost of the different schemes, as well as its sensitivity against different system parameters. We are also interested in cost components and distribution.
- Request performance: Finally, besides the cost performance, we also study the distribution of different types of requests/transmissions, i.e., direct hit rate (defined as the number of direct hit requests divided by total number of requests), differential transmission rate, indirect hit rate and replication miss rate.

## B. Preliminary Comparison Between Minimum Eviction and DPLO

Due to the scalability issues of *Minimum Eviction*, first consider a small scale problem with 3 servers, 14 movies and 3 chunks per movie for the comparisons in this sub-section. Fig. 6 shows the total switching cost as a function of the inter-view switch probability (view-switch tendency) for different schemes. A view-switch tendency of 0 means that users only perform temporal switch, and view-switch tendency of 1 means that users perform inter-view switch at every switch opportunity. *Super Optimal* is the optimal solution to the formulated ILP problem in Section IV but without the integer constraints. We observe that *Random* has the worst performance due to the lack of a problem-specific optimization. When view switch tendency is larger than 0, *DPLO* outperforms *Local Greedy* because the servers consider view switches when they

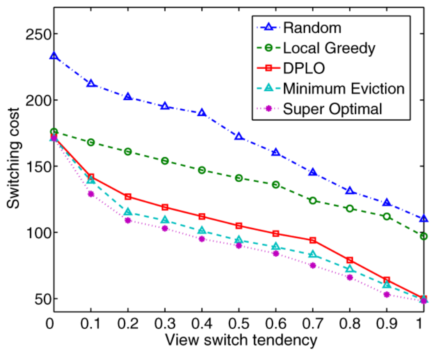


Fig. 6. Switching cost versus view-switch tendency for different replication algorithms.

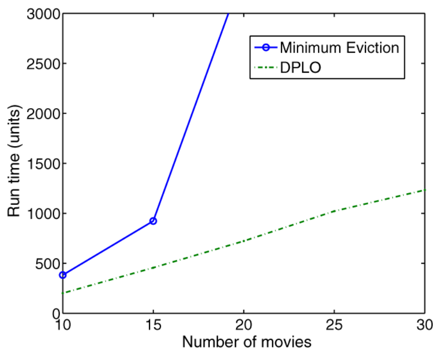


Fig. 7. Algorithm run time versus number of movies.

replicate chunks. Both *DPLO* and *Minimum Eviction* achieve close-to-optimal performance as compared to *Super Optimal*.

Fig. 7 shows the running time of *DPLO* and *Minimum Eviction* as a function of number of movies. *DPLO* achieves much better performance in run time than *Minimum Eviction*. This is because *Minimum Eviction* needs to solve a large-scale LP problem with a large number of variables. Recall that the computational complexity of *DPLO* can be tuned using the rounding parameter $R$, which is set to 2 in this experiment.

Although *Minimum Eviction* achieve better performance in switching cost, it does not scale to larger problem sizes. Therefore we focus on *DPLO* in the rest of the evaluation studies.

## C. Computational Run Time

We now study the performance of DPLO in different experiments. Fig. 8 shows the run time of *DPLO* as a function of switching cost. It illustrates the "time-performance" tradeoff of our algorithm. We use a scaling parameter $R$ to control the size of the DP table, and hence the complexity of the algorithm. As shown in the figure, with $R = 1$ (no rounding operations), *DPLO* gives the most accurate replicating solution, which leads to lowest switching costs. As the scaling factor $R$ increases, the run time of *DPLO* decreases significantly with a slight increase in switching costs. Therefore, by adjusting the scaling factor $R$, *DPLO* can trade off performance with computational complexity in the optimization of the replications strategy.

Fig. 9 shows the algorithm run time as a function of the total number of servers. It demonstrates the relationship between the computational time of a replication plan and the problem complexity. The run time of *Random* does not increase much with

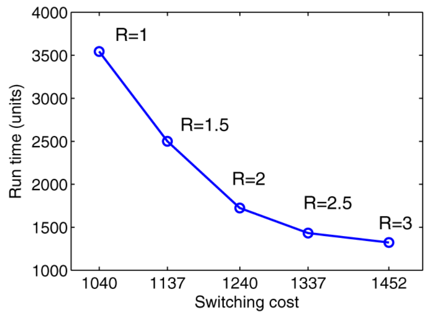


Fig. 8. Algorithm run time versus switching cost.

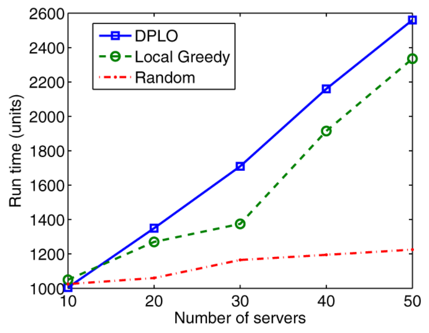


Fig. 9. Switching cost versus view-switch tendency for different replication algorithms.

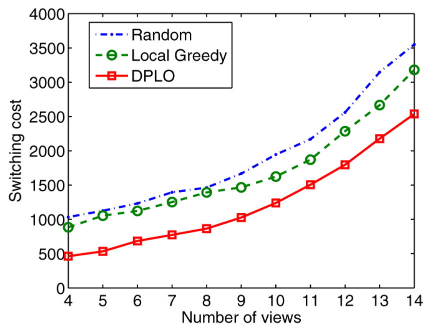


Fig. 10. Switching cost versus number of views for different replication algorithms.

the problem size since every server just randomly replicates chunks without any collaboration and optimization. *DPLO* and *Local Greedy* has similar performance, where their run time both increases with the number of servers. In real VOD systems, multiple servers are often grouped into a single server cluster or farm, which can be modeled as one logical server in our problem. Therefore the total number of servers is not expected to be very large.

## D. Switching Cost

Fig. 10 shows the switching cost as a function of the number of views in total. We observe that the switching cost increases with the number of views. This is because the increase in views leads to an increase of the number of total chunks in the system, and hence there is a higher likelihood of replication miss. We observe again that *DPLO* performs better than *Random* and *Local Greedy*, especially when the number of views is large.

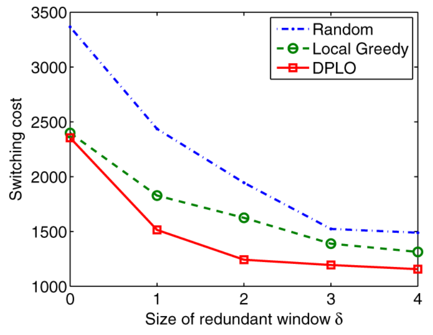


Fig. 11. Switching cost versus the size of redundant window for different replication algorithms.

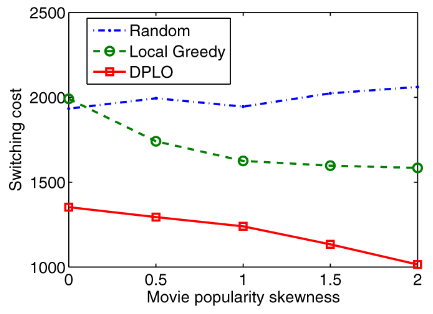


Fig. 12. Switching cost versus the movie popularity model for different replication algorithms.

Fig. 11 shows the switching cost as a function of the size of the redundant window $\delta$. The switching cost decreases with the increase of the redundant window size. With a larger redundant window, more view switches only require transmission of the corresponding pre-encoded differentials. As a result, the indirect hit rate increases, and the switching cost in turn decreases. On the other hand, a large redundant window size means that there are more redundant inter-view P-frames generated in the head of each coding unit, leading to a more redundant representation and to a larger storage cost at the repository. Therefore, the size of the redundant window needs to be judiciously selected to trade off performance (switching cost) with repository storage cost.

Fig. 12 shows the switching cost as a function of the Zipf parameter in the movie popularity model. When the parameter is equal to zero, all movies are equally popular. As the movie popularity becomes more skewed, the switching cost of *DPLO* and *Local Greedy* decreases because they use relatively more storage to replicate popular movies. The switching cost of *Random* is not sensitive to movie popularity. We observe again that *DPLO* significantly outperforms *Local Greedy* and *Random*.

## E. Cost Components and Distribution

Fig. 13 shows the cost distribution of the different schemes. It can be seen that *DPLO* has a much lower replication miss cost compared to *Random* and *Local Greedy*. This is because *DPLO* replicates chunks in order to maximize the benefit of transmitting frame differentials instead of the whole chunks from the repository. Therefore, there are much more differential transmission requests and indirect hits in *DPLO* than in the other two schemes. DPLO exploits indirect hit to lower the overall cost.

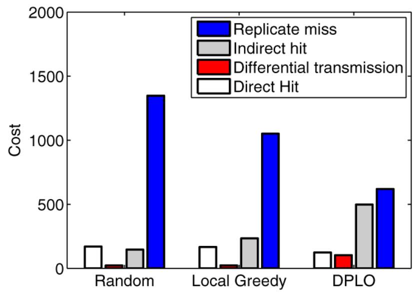


Fig. 13. Cost distribution for the different replication schemes.

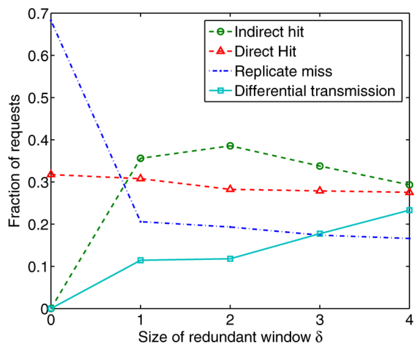


Fig. 14. Distribution of request types versus the size of redundant window for algorithm DPLO.

Fig. 14 shows the fraction of different requests in *DPLO* as a function of the size of the redundant window. When the redundant window size is zero, there are only two types of requests—direct hit and replication miss—since there is no inter-view P-frames to provide representation redundancy and exploit inter-view correlation during a view-switch. With the increase of the redundant window size, the fraction of replication miss decreases sharply, and both the fractions of indirect hit and differential transmission increase. The servers use inter-view P-frames to decode neighboring views more often, and the cost induced by replication miss significantly decreases. When the redundant window size becomes large, the fraction of indirect hits also decreases. This is because, with a large redundant window size, each view has a larger number of neighboring views. Hence most of the view switch requests lead to differential transmission, which costs less than an indirect hit. The number of direct hits remains constant, since the redundant window size does not introduce new replicated chunks.

## VIII. Conclusion

In interactive multiview video streaming (IMVS), users watching a multiview video may request inter-view or temporal switches at any time. In this paper, we study the issues of coding structure and its replication to support large-scale IMVS service with distributed servers. In order to facilitate view-switching and storage, we propose a coding structure based on redundant P-frames and merge frames. Using the redundant frame structure, the switching cost of video segments can be substantially reduced via "indirect hit"—given a requested view and a locally stored correlated views at a server, only pre-encoded frame differentials between the replicated views and the requested views are needed to be transmitted in the network.

With the coding structure, we then formulate the content replication problem to minimize content switching cost as an integer linear programming (ILP) problem. We propose an LP-based strategy with integer rounding (called Minimum Eviction) to replicate movie contents which achieves excellent performance. We further propose a more scalable solution based on dynamic programming and Lagrangian optimization (DPLO). Simulation results show that our schemes achieve very close to the optimal solution, with significantly lower cost than a state-of-the-art and a commonly used replication schemes.

## References

[1] M. Tanimoto, M. P. Tehrani, T. Fujii, and T. Yendo, "Free-viewpoint TV," *IEEE Signal Process. Mag.*, vol. 28, no. 1, pp. 67–76, Jan. 2011.

[2] G. Cheung, A. Ortega, and N.-M. Cheung, "Interactive streaming of stored multiview video using redundant frame structures," *IEEE Trans. Image Process.*, vol. 20, no. 3, pp. 744–761, Mar. 2011.

[3] J.-G. Lou, H. Cai, and J. Li, "A real-time interactive multi-view video system," in *Proc. ACM Int. Conf. Multimedia*, Singapore, Nov. 2005, pp. 161–170.

[4] X. Zhang and H. Hassanein, "Video on-demand streaming on the internet—a survey," in *Proc. 25th Biennial Symp. Commun.*, 2010, pp. 88–91.

[5] W. Dai, G. Cheung, N.-M. Cheung, A. Ortega, and O. Au, "Rate-distortion optimized merge frame using piecewise constant functions," in *Proc. IEEE Int. Conf. Image Process.*, Melbourne, Australia, Sep. 2013, pp. 1787–1791.

[6] P. Merkle, A. Smolic, K. Muller, and T. Wiegand, "Efficient prediction structures for multiview video coding," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 17, no. 11, pp. 1461–1473, Nov. 2007.

[7] S. Shimizu, M. Kitahara, H. Kimata, K. Kamikura, and Y. Yashima, "View scalable multiview coding using 3-D warping with depth map," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 17, no. 11, pp. 1485–1495, Nov. 2007.

[8] T. Fujii, K. Mori, K. Takeda, K. Mase, M. Tanimoto, and Y. Suenaga, "Multipoint measuring system for video and sound—100 camera and microphone system," in *Proc. IEEE Int. Conf. Multimedia Expo*, Toronto, ON, Canada, Jul. 2006, pp. 437–440.

[9] G. Cheung, A. Ortega, N.-M. Cheung, and B. Girod, "On media data structures for interactive streaming in immersive applications," in *Proc. SPIE Vis. Commun. Image Process. Conf.*, Huang Shan, China, Jul. 2010.

[10] X. Xiu, G. Cheung, and J. Liang, "Delay-cognizant interactive multiview video with free viewpoint synthesis," *IEEE Trans. Multimedia*, vol. 14, no. 4, pp. 1109–1126, Aug. 2012.

[11] T. Maugey, I. Daribo, G. Cheung, and P. Frossard, "Navigation domain representation for interactive multiview imaging," *IEEE Trans. Image Process.*, vol. 22, no. 9, pp. 3459–3472, Sep. 2013.

[12] M. Karczewicz and R. Kurceren, "The SP- and SI-frames design for H.264/AVC," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 13, no. 7, pp. 637–644, Jul. 2003.

[13] N.-M. Cheung, A. Ortega, and G. Cheung, "Distributed source coding techniques for interactive multiview video streaming," in *Proc. 27th Picture Coding Symp.*, Chicago, IL, USA, May 2009, pp. 1–4.

[14] E. Jaho, I. Koukoutsidis, I. Stavrakakis, and I. Jaho, "Cooperative content replication in networks with autonomous nodes," *Comput. Commun.*, vol. 35, no. 5, pp. 637–647, Mar. 2012.

[15] S.-H. G. Chan, "Operation and cost optimization of a distributed servers architecture for on-demand video services," *IEEE Commun. Lett.*, vol. 5, no. 9, pp. 384–386, Sep. 2001.

[16] S. Ataee, B. Garbinato, and F. Pedone, "Restream - a replication algorithm for reliable and scalable multimedia streaming," in *Proc. 21st Euromicro Int. Conf. Parallel, Distrib. and Network-Based Process.*, 2013, pp. 68–76.

[17] Y. Zhou, T. Fu, and D. M. Chiu, "On replication algorithm in P2P VoD," *IEEE/ACM Trans. Networking*, vol. 21, no. 1, 2013.

[18] S. Ghandeharizadeh and S. Shayandeh, "Domical cooperative caching for streaming media in wireless home networks," *ACM Trans. Multimedia Computing, Communications and Applications*, vol. 7, no. 4, pp. 40:1–40:17, Dec. 2011.

[19] S. Borst, V. Gupta, and A. Walid, "Self-organizing algorithms for cache cooperation in content distribution networks," *ACM SIGMETRICS Performance Eval. Rev.*, vol. 37, no. 2, pp. 71–72, 2009.

[20] S. Mao, X. Xheng, Y. T. Hou, H. D. Sherali, and J. H. Reed, "On joint routing and server selection for MD video streaming in ad hoc networks," *IEEE Trans. Wireless Commun.*, vol. 6, no. 1, pp. 338–347, Jan. 2007.

[21] M. Wang, L. Xu, and B. Ramamurthy, "Improving multi-view peer-to-peer live streaming systems with the divide-and-conquer strategy," *Comput. Netw.*, vol. 55, no. 18, pp. 4069–4085, Dec. 2011.

[22] S. Sedef Savas, C. Göktuğ Gürler, A. Murat Tekalp, E. Ekmekcioglu, S. Worrall, and A. Kondoz, "Adaptive streaming of multi-view video over p2p networks," *Image Commun.*, vol. 27, no. 5, pp. 522–531, May 2012.

[23] Y. Ding and J. Liu, "Efficient stereo segment scheduling in peer-to-peer 3D/multi-view video streaming," in *Proc. IEEE Int. Conf. Peer-to-Peer Computing*, Sep. 2011, pp. 182–191.

[24] Z. Chen, L. Sun, and S. Yang, "Overcoming view switching dynamic in multi-view video streaming over P2P network," in *Ptov. 3DTV-Conference: The True Vision - Capture, Transmission and Display of 3D Video*, Jun. 2010, pp. 1–4.

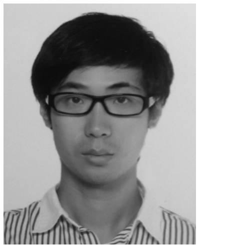
[25] H. Huang, B. Zhang, G. Chan, G. Cheung, and P. Frossard, "Coding and caching co-design for interactive multiview video streaming," in *Proc. Mini-conf. IEEE INFOCOM*, Orlando, FL, USA, Mar. 2011, pp. 3073–3077.

[26] H. Huang, S.-H. G. Chan, G. Cheung, and P. Frossard, "Near-optimal content replication for interactive multiview video streaming," in *Proc. 19th Int. Packet Video Workshop*, May 2012, pp. 95–98.

[27] C. H. Papadimitriou and K. Steiglitz, *Combinatorial Optimization: Algorithms and Complexity*. New York, NY, USA: Dover, 1998.

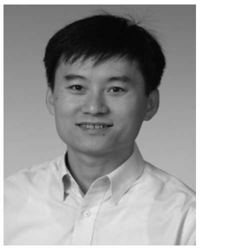
[28] Tanimoto Laboratory, Department of Information Electronics, Nagoya University [Online]. Available: http://www.tanimoto.nuee.nagoya-u.ac.jp

[29] S. Borst, V. Gupta, and A. Walid, "Distributed caching algorithms for content distribution networks," in *Proc. IEEE INFOCOM*, Mar. 2010, pp. 1–9.



**Dongni Ren** received the B.Eng. degree in computer science and engineering and M.Phil. degree in computer science from Hong Kong University of Science and Technology (HKUST), in 2007 and 2009, respectively, where he is currently working toward the Ph.D. degree at the Department of Computer Science and Engineering, supervised by Prof. Gary Chan.

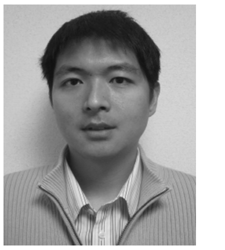
His research interest includes video streaming networks, overlay broadcasting, Video on Demand (VOD), and multi-view/free-viewpoint video technologies.



**S.-H. Gary Chan** (S'89–M'98–SM'03) received the B.S.E. degree (with highest honor) in electrical engineering from Princeton University, Princeton, NJ, USA, in 1993, and M.S.E. and Ph.D. degrees in electrical engineering from Stanford University, Stanford, CA, USA, in 1994 and 1999, respectively,

He is currently Professor and Undergraduate Programs Coordinator at the Department of Computer Science and Engineering, The Hong Kong University of Science and Technology (HKUST), Hong Kong. He is also the Director of Sino Software Research Institute at HKUST. His research interest includes multimedia networking, wireless networks and mobile computing.

Prof. Chan was an associate editor of the IEEE TRANSACTIONS ON MULTIMEDIA (2006–11), and a Vice-Chair of Peer-to-Peer Networking and Communications Technical Sub-Committee of IEEE Comsoc Emerging Technologies Committee. He has been Guest Editors of IEEE TRANSACTIONS ON MULTIMEDIA (2011), the *IEEE Signal Processing Magazine* (2011), the *IEEE Communication Magazine* (2007), and Springer Multimedia Tools and Applications (2007). He was the TPC chair of IEEE Consumer Communications and Networking Conference (CCNC) 2010, Multimedia symposium in IEEE Globecom (2007 and 2006), IEEE ICC (2007 and 2005), and Workshop on Advances in Peer-to-Peer Multimedia Streaming in ACM Multimedia Conference (2005). His research projects on wireless and streaming have received several ICT (Information and Communication Technology) awards in Hong Kong, Pan Pearl River Delta and Asia-Pacific regions due to their commercial impacts to industries (2012, 2013, and 2014). He is the recipient of Google Mobile 2014 Award (2010 and 2011) and Silver Award of Boeing Research and Technology (2009). He has been a visiting professor or researcher in Microsoft Research (2000–11), Princeton University (2009), Stanford University (2008–09), and University of California at Davis (1998–1999). He was a Co-director of HKUST Risk Management and Business Intelligence program (2011–2013), and Director of Computer Engineering Program at the HKUST (2006–2008).



**Gene Cheung** (M'00–SM'07) received the B.S. degree from Cornell University, Ithaca, NY, USA, in 1995, and the M.S. and Ph.D. degrees and computer science from the University of California, Berkeley, CA, USA, in 1998 and 2000, respectively, all in electrical engineering.

He was a Senior Researcher with Hewlett-Packard Laboratories Japan, Tokyo, from 2000 till 2009. He is now an associate professor in National Institute of Informatics in Tokyo, Japan. His research interests include image & video representation, immersive visual communication and graph signal processing. He has published over 130 international conference and journal publications.

He has served as an associate editor for the IEEE TRANSACTIONS ON MULTIMEDIA from 2007 to 2011 and currently serves as associate editor for DSP Applications Column in the *IEEE Signal Processing Magazine* and APSIPA journal on signal & information processing, and as area editor for *EURASIP Signal Processing: Image Communication*. He currently serves as member of the Multimedia Signal Processing Technical Committee (MMSP-TC) in IEEE Signal Processing Society (2012–2014). He has also served as area chair in IEEE International Conference on Image Processing (ICIP) 2010, 2012–2013, technical program co-chair of International Packet Video Workshop (PV) 2010, track co-chair for Multimedia Signal Processing track in IEEE International Conference on Multimedia and Expo (ICME) 2011, symposium co-chair for CSSMA Symposium in IEEE GLOBECOM 2012, and area chair for ICME 2013. He was invited as plenary speaker for IEEE International Workshop on Multimedia Signal Processing (MMSP) 2013 on the topic "3-D visual communication: media representation, transport and rendering". He is a co-author of best student paper award in IEEE Workshop on Streaming and Media Communications 2011 (in conjunction with ICME 2011), Best Paper finalists in ICME 2011 and ICIP 2011, Best Paper Runner-up Award in ICME 2012, and Best Student Paper Award in ICIP 2013.

**Pascal Frossard** (S'96–M'01–SM'04) received the M.S. and Ph.D. degrees, both in electrical engineering, from the Swiss Federal Institute of Technology (EPFL), Lausanne, Switzerland, in 1997 and 2000, respectively.

Between 2001 and 2003, he was a member of the research staff at the IBM T. J. Watson Research Center, Yorktown Heights, NY, where he worked on media coding and streaming technologies. Since 2003, he has been a faculty at EPFL, where he heads the Signal Processing Laboratory (LTS4). His research interests include image representation and coding, visual information analysis, distributed image processing and communications, and media streaming systems.

Dr. Frossard has been the General Chair of IEEE ICME 2002 and Packet Video 2007. He has been the Technical Program Chair of IEEE ICIP 2014 and EUSIPCO 2008, and a member of the organizing or technical program committees of numerous conferences. He has been an Associate Editor of the IEEE TRANSACTIONS ON IMAGE PROCESSING (2010–2013), the IEEE TRANSACTIONS ON MULTIMEDIA (2004–2012), and the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY (2006–2011). He is the Chair of the IEEE Image, Video and Multidimensional Signal Processing Technical Committee (2014–2015), and an elected member of the IEEE Visual Signal Processing and Communications Technical Committee (2006–) and of the IEEE Multimedia Systems and Applications Technical Committee (2005–). He has served as Steering Committee Chair (2012–2014) and Vice-Chair (2004–2006) of the IEEE Multimedia Communications Technical Committee and as a member of the IEEE Multimedia Signal Processing Technical Committee (2004–2007). He received the Swiss NSF Professorship Award in 2003, the IBM Faculty Award in 2005, the IBM Exploratory Stream Analytics Innovation Award in 2008 and the IEEE TRANSACTIONS ON MULTIMEDIA Best Paper Award in 2011.