Coding and Replication Co-Design for Interactive Multiview Video Streaming

Huan Huang Bo Zhang S.-H. Gary Chan
Dept. of Comp. Sci. & Eng.
The Hong Kong University of Science and Technology
Clear Water Bay, Hong Kong
{huangzunhuan, zhangbo, gchan}@cse.ust.hk

Gene Cheung
National Institute of Informatics
2-1-2 Hitotsubashi, Chiyoda-ku
Tokyo, Japan 101-8430
cheung@nii.ac.jp

Pascal Frossard
École Polytechnique Fédérale de Lausanne
EPFL-STI-IEL-LTS4, Station 11
CH-1015 Lausanne, Switzerland
pascal.frossard@epfl.ch

Abstract—**Multiview video refers to the simultaneous capturing of multiple video views with an array of closely spaced cameras. In an interactive multiview video streaming (IMVS) system, a client can play back the content in time in a single view, and may observe a scene of interest by switching to different viewpoints. Users independently choose their own view navigation paths through the high-dimensional multiview data. Distributed servers are deployed to collaboratively replicate video content in order to support user scalability. Such a system typically presents challenges in both *coding* and content *replication*. In coding, the multiview video must be encoded in order to support efficient view-switching and distributed replication. In content replication, it is important to decide which data blocks to store at each server to facilitate view-switches at any time.**

In this paper, we *co-design* a coding structure and a distributed content replication strategy. First, we propose a coding structure based on redundant P-frames and distributed source coding (DSC) frames to achieve efficiency in coding, view switches and content replication. We then propose a heuristic-based distributed and cooperative replication strategy to take advantage of the correlation between the multiple views for resource-effective content delivery. Simulation results show that our coding and replication co-design is cost-effective in supporting IMVS services.

I. Introduction

Multiview video refers to the simultaneous capturing of multiple videos of the same scene of interest by a large array of closely spaced cameras from different viewpoints [1]. In interactive multiview video streaming (IMVS) services [2], a client can play back the captured multiview video content in time in a single view, and may switch at any time to new views in order to observe a scene of interest from different viewing angles. Active selection of viewpoints by a client can engender a depth perception in the observed 3D scene [3]. It also represents a fundamental departure from traditional single-view video streaming, where media interaction is limited to basic and relatively infrequently used operations along the time dimension: pause, play, fast forward and rewind. Due to its compelling visual experience, multiview video has wide applications in education, entertainment, surveillance, health care, etc.

In order to serve distributed users and reduce access costs, a distributed servers architecture may be used [4]. In such a network, local servers are set up close to user pools. A remote repository with limited bandwidth stores the entire original pre-encoded multiview data. A client[1] is associated with a local server, and sends interactive requests to the server. The server serves it directly if the requested data has been stored; otherwise, it identifies a remote server or repository to supply the missing data (with some cost).

The design of cost-effective IVMS systems relies on appropriate coding and data replication strategies. In *coding*, the multiview video content must be pre-encoded at generation time, so that at streaming time, pre-encoded frames corresponding to clients' chosen viewpoints can be efficiently extracted for decoding and display. A simple (but naive) way to support view-switching at streaming time is to encode images of different viewpoints as independently coded I-frames; in this case, selected views can be transmitted and decoded in any order. However, because I-frames are not coding-efficient, this results in high bandwidth requirements during interactive view switches. Due to the high correlation among adjacent viewpoint images (because of the spatial proximity of capturing cameras), more efficient encoding can be proposed for good compression ratio and decoding flexibility.

In content *replication*, because each IMVS client independently selects only a small subset of frames out of the very large multiview video content during his view navigation, it is not possible to store a priori all the possible requests in the local servers. If the server deploys a naïve replication strategy of storing only the most popular

This work was supported, in part, by the General Research Fund from the Research Grant Council of the Hong Kong Special Administrative Region, China (611209) and Proof-of-Concept Fund at the HKUST (PCF.005.09/10 & PCFX05-15C00610/11ONA).

[1]In this work, we use "client" and "user" interchangeably.

multiview data, the hit ratio will be low and the network would not be efficient, because view navigation paths are typically unique and non-overlapping..

In this paper, we study how to *co-design* both the coding structure and the replication strategy to support efficient view-switching for IMVS services. First, in order to facilitate view-switching and storage, we propose a coding structure based on redundant P-frames and distributed source coding (DSC) frames to help users navigate among adjacent views [2], [5]. Given this novel structure, we then propose an efficient and cooperative heuristic called MVCR (Multiview Video Cooperative Replication) to efficiently replicate views. MVCR is fully distributed and guarantees convergence. In MVCR, even if an exact requested video view in local servers is not stored (i.e., *direct hit*), a different but correlated video view can nonetheless be requested from neighboring servers (*indirect hit*) so that access costs are reduced compared to requesting views from the main server. We show by simulation results that the proposed co-design solution leads to interactive content delivery with effective use of network resources.

The paper is organized as follows. After discussing related work in Section II, we present in Section III the IMVS network and coding structure to support efficient view-switching. The MVCR algorithm is discussed in Section IV. We present simulation results in Section V, and conclude in Section VI.

II. Related Work

Research in multiview video coding (MVC) has been mostly focused on compression of all captured frames, exploiting both temporal and inter-view correlation to achieve coding gain [6]. Though suitable for compact storage of all multiview data (e.g., on a DVD disc), for IMVS application [2] where only a single requested view per client is needed at one time, complicated inter-frame dependencies among coded frames across time and view reduce the random decodability of the video stream. In contrast, we propose an efficient frame structure where an image can be encoded into multiple versions, so that the appropriate version can be transmitted depending on the available content in decoder's buffer at stream time, in order to reduce server transmission rate.

Optimal replication in a distributed video-on-demand network is generally regarded as NP-hard (see, for example, [7]). We propose and study an efficient heuristic to address that. Previous work on replication in video streaming has focused only on single view videos, where there is no correlation among different streams [4], [8]–[10]. Furthermore, the replication strategy in this body of work has not considered the cooperation among neighboring servers to achieve low access cost. Our work differs from them by considering *jointly* the unique coding structure to support static-view switching for IMVS, and how to optimally replicate the data blocks to achieve low access costs.

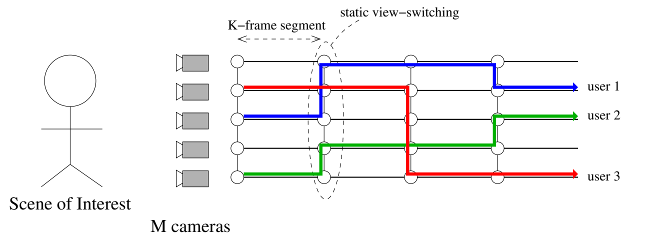

Fig. 1. In an IMVS scenario with static view-switching, different users select different view trajectories in the high-dimensional media space.

III. IMVS System and Coding Structure

We first overview the IMVS system under consideration, and then describe the type of media interaction (static view-switching) we provide for each client. We then propose a coding-efficient multiview frame structure by exploiting the inter-view correlation among neighboring viewpoints while enabling decoding flexibility in an IMVS session.

Each client is connected to a local server which provides IMVS service to the client, i.e., the server transmits multiview data corresponding to the client's interactively chosen view navigation path. If the requested data has already been stored in the server, the server will directly forward data to the client. If not, the server will identify another server or remote repository to satisfy the view request. Given the access popularity, each server hence has to decide which video views of what time instants to store given its limited storage in order to achieve overall low access cost.

We consider that a user can play back video of the *same* view for K consecutive frames in time; such K frames of the same view is called a *coding unit*. At the first frame or *head* of a coding unit, a user can interactively select different views of the same time instant (first frames of other coding units) before deciding on one view to resume video playback. Because there is no time progression during this interactivity, we call this navigation of views of the same time instant *static view-switching* (called *frozen moment* in [11]). See Fig. 1 where users 1, 2 and 3 choose different view navigation paths in a 5-camera multiview video sequence.

To support static view-switching while achieving good compression efficiency, we propose the following frame structure to pre-encode a given multiview video content. The video sequence of view i, $i = 1, \ldots, M$, is encoded into coding units, each K consecutive pictures long; i.e., a coding unit $B_{n,i}$ of view i, $n \geq 0$, contains the leading picture $F_{nK,i}$ of instant nK of view i, and trailing $K-1$ pictures $F_{nK+1,i}, \ldots, F_{(n+1)K-1,i}$. The leading picture $F_{nK,i}$ (head of coding unit $B_{n,i}$) serves for static view navigation and has a *redundant representation*, so that inter-view correlations among adjacent viewpoints are exploited.

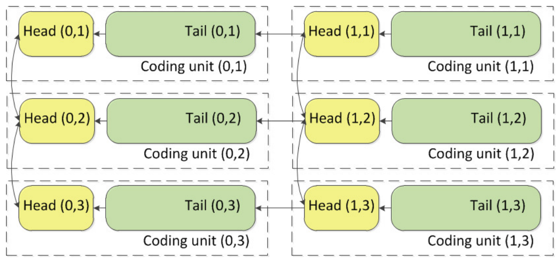

Fig. 2. Dependencies among coding units in proposed multiview video frame structure. Arrows among heads indicate feasible view switches using pre-encoded differentials.

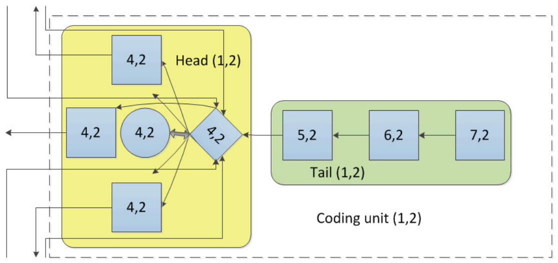

Fig. 3. An implementation of a coding unit using I-, P- and DSC frames (denoted by circle, squares, and diamonds, respectively).

Specifically, the head $F_{nK,i}$ has pre-encoded differentials in the structure to heads $F_{nK,i\pm1}$'s, so that a view-switch from $F_{nK,i}$ to $F_{nK,i\pm1}$ only requires transmission of the differential. See Fig. 2 for an illustration of a multiview video frame structure for $M = 3$.

In more details, the head of coding unit $B_{n,i}$ 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 *distributed source coding* (DSC) frame $W_{nK,i}$. Each P-frame $P_{nK,i}(i \pm 1)$ is predicted from a I-frame $I_{nK,i\pm1}$ of an adjacent view $i\pm1$. With I-frame $I_{nK,i}$ as coding target, DSC frame $W_{nK,i}$ uses P-frames $P_{nK,i}$'s as predictors; by DSC's construction [5], only *one* correctly decoded predictor frame is needed for correct DSC decoding. Thus, by transmitting the DSC frame $W_{nK,i}$ in combination with any one of the P-frames $P_{nK,i}$'s, $W_{nK,i}$ can be correctly decoded to reconstruct the target frame $I_{nK,i}$. In essence, $W_{nK,i}$ is a *merge* operator to reconciliate minor differences due to motion compensation and quantization among all possible P-frames $P_{nK,i}$'s. Figure 3 shows an illustration of frame structure for coding unit $(1, 2)$.

IV. MVCR: Multiview Video Cooperative Replication

In this section, we consider that there is certain access cost depending on data size between any pair of servers (including repository). Given the coding structure, we study which views to store in each server to cooperatively minimize the overall network access cost.

We consider that the coding units are replicated independently and focus on a static view-switching moment/instant when *view* interaction occurs (i.e., no time dimension of the video is considered). In this case, it suffices to focus on the replication of the *head* of the unit of a certain view-switching instant. For simplicity, we consider that servers store only I-frames (A server's decision to replicate view j means that it pulls I-frame I_j from the repository to its local storage *a priori*.) To support efficient view browsing at an arbitrary view-switching instant, each server allocates a certain replication capacity in terms of number of I-frames.

The view-switch cost has to be accounted for carefully. This cost depends on which one of the following four possible cases when a client at server x switches from view i to j that is not stored locally in the server:

1) *Direct hit*: This is the case when a neighboring server y has stored the exact view j, and hence the view can be forwarded to x directly.
2) *Differential transmission*: This is the case when the repository transmits pre-encoded differentially coded P-frame $P_j(i)$.
3) *Indirect hit*: This is the case when a neighboring server y has a *correlated* frame of view k, and hence that frame is forwarded to x and the repository transmits a differential $P_j(k)$ (as in the above case).
4) *Content miss*: This is the case when all the neighboring servers do not have exact or correlated frames, and the repository transmits independently-coded I-frame of view j to x.

At each server, users request view j with a certain probability π_j (view popularity). Given its limited capacity, each server replicates the views according to the popularity so that the replication leads to good view "locality" to support efficient switching with minimum overall view access cost. (Note that the "cost" is a general term that may be bandwidth access cost, user interactive delay, or combination of both.)

To address the replication problem, we propose an efficient heuristic called MVCR (Multiview Video Cooperative Replication) for each server to decide which view to replicate and which neighbors to solicit help upon a view switch. MVCR is fully distributed and guarantees convergence. It continuously reduces the overall access cost through message exchanges with neighboring servers.

In MVCR, each server adopts a *probabilistic* server selection strategy as follows. Let N_j^x be the set of neighbors of server x that store view j. To reduce the access cost for view j, the server x (without the view j) chooses a server y with the view according to the access cost $t_j^{x,y}$ with a certain probability. This probability is given by

$$H_j^{x,y} = \frac{t_j^{x,y}}{\sum_{y \in N_j^x} t_j^{x,y}}. \tag{1}$$

To decide whether to replicate view j at server x, the server needs to consider the two cases of direct and indirect hit. In direct hit, the server x requests from server y view j from server y with a probability $H_j^{x,y}$ given in Equation (1). Let $C_j^x(1)$ be the average cost to access view j due to direct hit at server x. We have

$$C_j^x(1) = \sum_{y \in N_j^x} H_j^{x,y} t_j^{x,y} \pi_j. \quad (2)$$

For indirect hit, the neighboring servers of server x can supply view k while the repository supplies the pre-encoded differential, *provided that* $|j - k| \leq 1$. Let $C_j^x(3)$ be the average cost to access view j from the neighboring servers of server x due to indirect hit. We have

$$C_j^x(3) = \sum_{k \in j \pm 1} \pi_k \left(\sum_{y \in N_k^x} H_k^{x,y} t_k^{x,y} + t_{k,j}^{x,R} \right). \quad (3)$$

Let C_j^x be the average cost to access view j at server x, which is clearly

$$C_j^x = C_j^x(1) + C_j^x(3). \quad (4)$$

Let $C_j^{xy^+}$ and $C_j^{xy^-}$ be the average cost at server x to access view j from server y with and without the view, respectively ($C_j^{xy^+} \leq C_j^{xy^-}$). Let J_j^{xy} be the change of such costs, i.e.,

$$J_j^{xy} = C_j^{xy^-} - C_j^{xy^+}. \quad (5)$$

Equipped with the above notation, our MVCR replication strategy is finally stated as follows. When a server x makes its replication decision for view k, it collects from each of its neighbors, say server y, its cost change J_k^{yx}. It then sums these changes together and estimates A_k^x. Specifically, if N^x is the set of neighboring servers of server x, the server x computes

$$A_k^x = \sum_{y \in N^x} J_k^{yx}. \quad (6)$$

The server x then replaces view k with view k' if $(A_k^x - A_{k'}^x)/\sum_k A_k^x$ is larger than a certain threshold α ($\alpha < 1$).

V. Simulation Environment and Illustrative Results

We study MVCR performance using simulation. We use the well-known MPEG multiview video test sequence `pantamime` of resolution 1280×960. Views of `pantamime` are coded into our proposed frame structure using a H.263+ codec: I- and P-frames are coded using conventional H.263+ tools, while DSC frames are coded using codec in [5] that was built using H.263+ tools with added modifications. The quantization parameter is fixed at a constant for all frames for constant visual quality. The view size is the size of the I-frame. We conduct simulations on a real Internet topology taken from [www.caida.org], which contains 1,747 nodes and 3,732 links each with a certain delay.

We consider user interactivity where view-switch requests of a certain head may be modeled by a recurrent

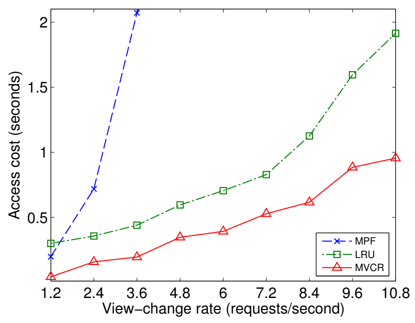

Fig. 4. Access delay versus request rate given different schemes.

DTMC (Discrete-time Markov chain). The state is represented by $\{1, 2, \ldots, M\}$, which is the view index browsed by the user. There is an anchor or default view indexed by m, from which a user may jump to any other view with probability $p_{m,j}$ ($j \neq m$). For other states, a user may jump back to the anchor view, or move up or down a view with certain probabilities. In our simulations, we use $p_{1,m} = 3/4, p_{i,i+1} = 1/4, p_{i,i-1} = 1/4, p_{M,m} = 3/4$, $p_{i,m} = 1/2$, and $p_{m,m\pm j} \propto 1/j^s$ (a Zipf distribution), where $j \geq 1$. Due to such transition probability, we consider at steady state that each server experiences view-switch requests according to a Poisson process with rate λ (/second).

We define the transmission cost as the total user interactive delay, which is the sum of the delay of all the links that connect the two servers, with each link delay modeled by M/M/1. If the request cannot be served immediately because of a lack of available bandwidth, it will wait till resources are sufficient. We further consider that images are downloaded to the server completely before they can be browsed (or played back).

Unless stated otherwise, we use the following baseline (default) parameters in simulations: link bandwidth of 1 Gbits/s, number of servers equal to 30, number of views set to 100, storage capacity of each server equal to 5, $s = 0.6$, view downloading bitrate fixed at 5 Mbits/s, view size of 500 kbytes, replicate and replacement interval of 4 minutes, $\lambda = 6$ requests/second.

The performance metrics that we are interested in is the access cost (or delay) for all served requests, which is defined as the delay from the time of request till the time of downloading the view from a server or from the repository. This delay is hence the sum of time waiting for available network bandwidth and time for downloading the view completely.

We compare MVCR with the following replication schemes:

- *LRU (Least Recently Used)*, where each server replaces the least-recently used view with the most recently requested one;
- *MPF (Most Popular First)*, where each server replicates the most popular views.

We plot in Figure 4 the access cost versus λ for

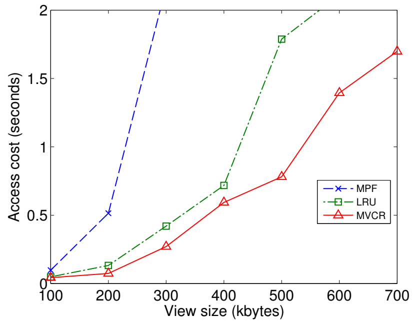

Fig. 5. Access cost versus view size given different schemes.

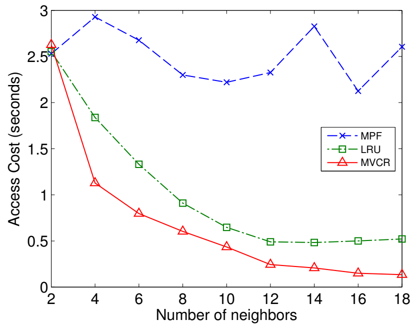

Fig. 6. Access cost versus number of neighbors size given different schemes.

different schemes. The access cost increases with λ because of larger network traffic leading to congestion. MVCR has the lowest access cost compared to the other schemes. It balances the traffic in the network during server selection and appropriate replication, leading to a lower access cost and better server load balancing. Given a certain cost constraint, MVCR can accommodate much higher request rates. LRU does not perform well because it continuously replaces its storage, leading to high replacement traffic in the network. MPF performs the worst because all the servers replicate only the hottest views. This leads to poor collaboration between servers, and hence high bandwidth and storage utilizations. Its high access cost is due to congestion at the repository. The figure shows that it is important to consider the cost change due to replication of neighboring servers and cooperative replication.

Figure 5 plots the access cost versus the view size for difference schemes. Access cost increases with the view size due to higher downloading cost/time, and a longer waiting time for bandwidth availability. MVCR has much lower access cost as compared to other schemes. Given a constraint on the access cost, MVCR can support significantly larger view sizes. This leads to a much higher view quality.

We show in Figure 6 the access cost versus the number of neighboring servers. MPF is not so sensitive to the number of neighbors, because all the neighbors cache the same views and hence the server collaboration is minimal. For the other schemes, the access cost decreases with the number of servers because of higher better server collaboration and better traffic spreading among them. MVCR performs best because it attains higher collaboration among neighboring servers through better view replication.

VI. Conclusion

In this paper, we consider the coding and content replication co-design problem for a network with distributed servers to support interactive multiview video streaming services (IMVS). We present an efficient coding structure using redundant P-frames and distributed source coding (DSC) frames that facilitates view switching and supports distributed replication. With this coding structure, we propose a novel cooperative replication scheme called MVCR (Multiview Video Cooperative Replication) to effectively store views at each server. MVCR is fully distributed, performs efficiently and guarantees convergence. We have conducted extensive simulations with a realistic Internet topology to study the performance of MVCR. Our results show that MVCR outperforms other schemes by a wide margin in terms of access cost/delay.

References

[1] A. Kubota, A. Smolic, M. Magnor, M. Tanimoto, T. Chen, and C. Zhang, "Multi-view imaging and 3DTV," in *IEEE Signal Processing Magazine*, vol. 24, no.6, November 2007.

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

[3] C. Zhang, Z. Yin, and D. Florencio, "Improving depth perception with motion parallax and its application in teleconferencing," in *IEEE International Workshop on Multimedia Signal Processing*, Rio de Jeneiro, Brazil, October 2009.

[4] S.-H. Chan and F. Tobagi, "Distributed servers architecture for networked video services," *IEEE/ACM Transactions on Networking*, vol. 9, no. 2, pp. 125–136, Apr 2001.

[5] N.-M. Cheung, A. Ortega, and G. Cheung, "Distributed source coding techniques for interactive multiview video streaming," in *27th Picture Coding Symposium*, Chicago, IL, May 2009.

[6] M. Flierl, A. Mavlankar, and B. Girod, "Motion and disparity compensated coding for multiview video," in *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 17, no.11, November 2007, pp. 1474–1484.

[7] A. Nimkar, C. Mandal, and C. Reade, "Video placement and disk load balancing algorithm for VoD proxy server," in *Internet Multimedia Services Architecture and Applications (IMSAA), 2009 IEEE International Conference on*, Dec. 2009, pp. 1–6.

[8] Y. R. Choe, D. L. Schuff, J. M. Dyaberi, and V. S. Pai, "Improving VoD server efficiency with bittorrent," in *Proceedings of the 15th international conference on Multimedia*, New York, NY, USA, 2007, pp. 117–126.

[9] W.-P. K. Yiu, X. Jin, and S.-H. G. Chan, "VMesh: Distributed segment storage for peer-to-peer interactive video streaming," *IEEE Journal on Selected Areas in Communications (JSAC) special issue on Advances in Peer-to-Peer Streaming Systems*, vol. 25, no. 9, pp. 1717–31, Dec. 2007.

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

[11] J.-G. Lou, H. Cai, and J. Li, "A real-time interactive multi-view video system," in *ACM International Conference on Multimedia*, Singapore, November 2005.