# Collaborative Wireless Freeview Video Streaming With Network Coding

Bo Zhang, *Student Member, IEEE*, Zhi Liu, *Member, IEEE*, S.-H. Gary Chan, *Senior Member, IEEE*, and Gene Cheung, *Senior Member, IEEE*

***Abstract*—Free viewpoint video (FVV) offers compelling interactive experience by allowing users to switch to any viewing angle at any time. An FVV is composed of a large number of camera-captured anchor views, with virtual views (not captured by any camera) rendered from their nearby anchors using techniques such as depth-image-based rendering (DIBR). We consider a group of wireless users who may interact with an FVV by independently switching views. We study a novel live FVV streaming network where each user pulls a subset of anchors from the server via a primary channel. To enhance anchor availability at each user, a user generates network-coded (NC) packets using some of its anchors and broadcasts them to its direct neighbors via a secondary channel. Given limited primary and secondary channel bandwidths at the devices, we seek to maximize the received video quality (i.e., minimize distortion) by *jointly* optimizing the set of anchors each device pulls and the anchor combination to generate NC packets. To our best knowledge, this is among *the first body of work* addressing such joint optimization problem for wireless live FVV streaming with NC-based collaboration. We first formulate the problem and show that it is NP-hard. We then propose a scalable and effective algorithm called PAFV (Peer-Assisted Freeview Video). In PAFV, each node collaboratively and distributedly decides on the anchors to pull and NC packets to share so as to minimize video distortion in its neighborhood. Extensive simulation studies show that PAFV outperforms other algorithms, achieving substantially lower video distortion (often by more than 20–50%) with significantly less redundancy (by as much as 70%). Our Android-based video experiment further confirms the effectiveness of PAFV over comparison schemes.**

***Index Terms*—Distributed computing, multimedia computing, wireless networks.**

## I. Introduction

FREE viewpoint video (FVV) is a novel video format which enables new and compelling viewing experience by allowing the user to choose any angle to view at any time when the video is being played [1], [2]. For example, a laboratory experiment or a musical instrument lesson can be better understood if an audience can observe the video by switching to any viewing angles at any time.

In an implementation of FVV, *anchor views* of a scene are captured simultaneously in texture-plus-depth format by a discrete array of cameras. A user receiving some of these anchors may either play back an anchor, or render a *virtual view* (i.e., views not captured by any camera) given his/her available anchors using Depth-Image-Based Rendering (DIBR) [3]. The quality of the rendered view depends on the spatial correlation between the view and the available anchors. Such quality decreases with the distance from adjacent anchors [4].

We consider a group of wireless users interested in a live FVV. They pull the anchor streams from a remote server (through, for example, 4G data network). Due to the interactive nature of FVV, though all the users view the same video, their viewing angles may be different (asynchronous) and may change over time. As wireless bandwidth is precious, how to efficiently utilize the bandwidth to provide the best possible received video quality to the users becomes a challenging problem.

One simple approach is that, upon each view switch at a user, the server renders the requested view for the user.[1] This approach, however, would easily overwhelm the server with frequent view switch requests and view computation. Another approach is to stream all the anchors to users for them to render all the views they need. However, due to the large number of anchors (usually tens to hundreds), this would consume prohibitively large server and network bandwidths. To relieve server and network bandwidths, each user may pull only a subset of anchors and render any desired view from it, but this leads to severely degraded video quality due to increased anchor distances.

Wireless devices nowadays are often equipped with multiple network interfaces. We therefore consider a novel network that utilizes these interfaces for live FVV streaming based on network coding and anchor sharing. In this network, each user collaboratively pulls a subset of anchors from the server via a *primary* channel (e.g., cellular data network), and shares some of the anchors by *broadcasting* via a *secondary* channel (e.g., Wi-Fi Direct).[2] Using the pulled and neighbor-shared anchors, each user can then render any desired view locally.

To enhance sharing efficiency, we use linear network coding (NC) for anchor sharing. A user selects some of its pulled

Manuscript received August 12, 2015; revised November 29, 2015; accepted January 2, 2016. Date of publication January 18, 2016; date of current version February 18, 2016. This work was supported in part by the Hong Kong Research Grant Council (RGC) General Research Fund under Grant 610713, in part by the Hong Kong Innovation and Technology Fund under Grant UIM/246, and in part by the JSPS KAKENHI under Grant 15K21599. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Shiwen Mao.

B. Zhang and S.-H. G. Chan are with the Department of Computer Science and Engineering, The Hong Kong University of Science and Technology, Hong Kong, China (e-mail: zhangbo@cs.ust.hk; gchan@cs.ust.hk).

Z. Liu is with the Global Information and Telecommunication Institute, Waseda University, Tokyo 169-8555, Japan (e-mail: liuzhi@aoni.waseda.jp).

G. Cheung is with the National Institute of Informatics, Tokyo 101-8430, Japan (e-mail: cheung@nii.ac.jp).

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.2016.2518485

[1]In this paper we use "user", "client", "node", and "peer" interchangeably.

[2]In this paper, "broadcast" refers to the mechanism of sending a single packet to cover all the neighbors. The "neighbors" are the reachable nodes by a single hop from a user.

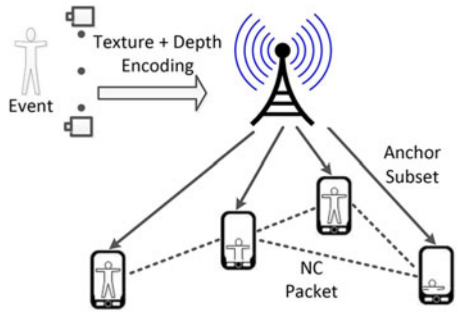


Fig. 1. Cooperative wireless live streaming network for texture-plus-depth coded FVV.

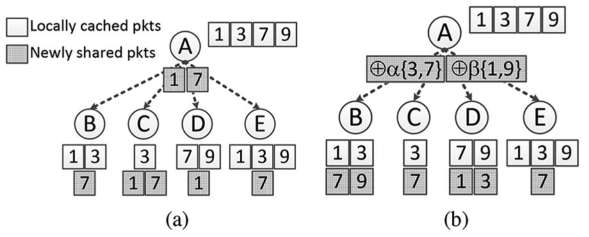


Fig. 2. Illustration of the benefit of network coding. (a) Without network coding. (b) With network coding.

anchors, codes their packets into some NC packets, and broadcasts them to its neighbors. Each user decodes the received NC packets to recover more anchors. This greatly enhances its anchor availability, leading to higher view quality.

We show in Fig. 1 a collaborative wireless network for live FVV streaming. Solid and dotted lines represent anchor subset pulling via the primary channel and NC packet sharing in the secondary channel, respectively. A live event is recorded by a camera array, each of which encodes the captured video stream (anchor) into the texture-plus-depth format. The encoded anchors are then streamed to a group of co-located mobile clients via a base station. Due to bandwidth limitations in wireless channels, each client receives a subset of anchors from the base station in the primary channel (solid lines) and collaboratively broadcasts in the secondary channel (dotted lines) the NC packets formed by some of the pulled anchors. In this way, a larger and more diverse set of anchors are available at each user, resulting in lower video distortion.

We show an example of anchor packet sharing in Fig. 2. Node $A$ has locally cached anchor set $\{1,3,7,9\}$, and is going to share some with its neighbors $B, C, D$ and $E$, each currently possessing anchor set $\{1,3\}, \{3\}, \{7,9\}$ and $\{1,3,9\}$, respectively. White rectangles indicate cached anchor packets at each node before $A$'s sharing, while shaded rectangles represent packets shared from $A$, as well as those that become available after sharing. Suppose $A$ has a secondary channel bandwidth of two packets. Without network coding as shown in Fig. 2(a), $A$ has to choose two out of the four anchor packets, say 1 and 7, to share with its neighbors. After such sharing, three anchor packets remain missing, namely anchor 9 at node $B$ and $C$, and anchor 3 at node $D$. With NC as shown in Fig. 2(b), $A$ codes two NC packets, one from $\{3,7\}$ with coefficient vector $\alpha$, the other from $\{1,9\}$ and coefficient vector $\beta$, and shares them to the neighbors. In this case, all but two missing packets can be recovered, with anchors 1 and 9 missing at $C$. NC clearly results in higher transmission efficiency. The *NC strategy*, i.e., the selection of participating data packets in each NC packet, directly affects the decodability, hence the transmission efficiency and video quality at each receiver.

We consider that the clients have heterogeneous primary and secondary bandwidths due to, for examples, wireless standards used, channel conditions, transmission power, etc. Each client needs to determine: 1) which anchors to pull from the server, and 2) which of the anchor packets should be used to code NC packets for sharing. Clearly, the two problems are interdependent, and hence need to be jointly optimized in order to minimize video distortion (in terms of Peak Signal-to-Noise Ratio (PSNR)). We will mainly focus on the rendering distortion of FVV due to view generation. The distortion caused by other factors (such as lossy compression or packet loss) has been considered independently by other literatures and may be treated as orthogonal to our study.

To the best of our knowledge, this is among the first body of work that studies the joint problem of anchor pulling and sharing for wireless live FVV streaming with NC-based cooperation, with 1-hop broadcast feature enabled. In summary, this paper makes the following key contributions.

1) *A collaborative wireless live FVV streaming network and its joint optimization problem:* We propose and study a wireless live FVV streaming network where clients collaboratively pull anchors and share NC packets formed by the anchors. We present a general middleware, residing between the FVV source and the FVV applications, to handle anchor distribution.
   We formulate the novel problem that *jointly* optimizes anchor pulling and NC generation in each client, in order to minimize the overall video distortion, subject to bandwidth constraints. We show that the problem is NP-hard.
2) *PAFV, distributed live FVV streaming with NC-based peer cooperation:* In a wireless network, it is difficult to centrally compute the optimal solution and timely distribute it to all the clients. We therefore propose a fully distributed, efficient and scalable algorithm called *PAFV* (Peer-Assisted Freeview Video). Each node independently decides which anchors to pull, and what anchors to use to code the NC packets for sharing, so as to aggressively reduce the overall video distortion in the neighborhood.
3) *Extensive simulations and experimental studies:* We conduct extensive simulations. Our simulation results show that PAFV achieves low overall received video distortion with low redundancy. PAFV is simple and effective. We have implemented PAFV on android devices with Wi-Fi Direct[3] to experimentally demonstrate the implementability and working of PAFV.

The remainder of this paper is organized as follows. We briefly discuss related work in Section II. We then introduce the

[3]"Wi-Fi Peer-to-Peer," [Online]. Available: http://developer.android.com/guide/topics/connectivity/wifip2p.html

middleware design and the problem formulation in Section III. We next describe PAFV in details in Section IV. Simulation results and experiment results are then presented in Sections V and VI, respectively. Finally we conclude in Section VII.

## II. Related Work

Wireless multi-hop live streaming for single-view videos has been extensively studied, with some focusing on minimizing delay [5]–[7], while others studying energy optimization [8]–[11]. There has also been work on optimizing the overlay structure or a multi-radio multi-channel network [12], [13]. Our work differs by studying a novel wireless live FVV streaming network with collaborative anchor pulling and NC generation, and its joint optimization problem.

Various coding techniques for multiview/freeview video have been proposed. In simulcast [14], views are treated independently as conventional videos. Multiview video coding (MVC) intends to exploit both temporal and spatial statistical correlations [15]–[18]. Although highly efficient, MVC is not easy to be extended for the bandwidth-limited wireless environment due to the strict coding dependency. MVC also restricts users to switch to only certain views at particular times. The FVV (texture-plus-depth format) we consider here offers much higher flexibility—DIBR is able to render virtual view of *any* angle with *any* subset of anchors at *any* time [3].

FVV is a novel and emerging video format, with most recent work focusing on improving the coding efficiency [19]–[22]. The work in [4] has extensively studied the distortion of a rendered view as a function of its distances from available anchors. There has been little research effort put into its wireless live streaming. In [1], a cloud is employed to compute and render each requested view. The work in [23] studies view-switching prediction to reduce traffic and computation load at the server for each view switch. As each view switch generates traffic and computation load at the server, the work in [23] focuses on reducing such load by view-switching prediction. These works are independent of, and may be integrated with, ours to achieve higher efficiency.

There has been little work studying live FVV streaming in wireless networks. The work in [24] uses unicast pull-based transmission with receiver-initiated NC, where each user computes the optimal NC that it needs from its parent, and pulls them from the parent. Due to its unicast nature, this may result in high traffic redundancy and control overhead which consume high parent bandwidth. PAFV instead utilizes push-based broadcast for high efficiency. For rendering a virtual view, [24] requires anchors of both left and right sides of the target view. We relieved such requirement in PAFV. These differences lead to drastically different anchor pulling and NC generation decisions.

In the linear network coding adapted in PAFV, source packets of an anchor subset are each multiplied by a coefficient randomly drawn from a finite field before transmission. This is different from, and more efficient than conventional random linear network coding (RLNC) in that we allow a subset of anchors be selected to generate a NC packet. This greatly facilitates and enhances partial recovery. The work in [25], [26] both consider a general cooperative recovery scenario for full recovery, but has not considered the joint problem of packet pulling and sharing. They employ Instantly Decodable Network Coding (IDNC) with *xor* operation, which only considers the decodability of individual NC packets. The work has not considered virtual view rendering characteristics in FVV streaming. We instead study a live FVV streaming network, and its *joint* optimization of anchor pulling and NC generation to minimize view distortion. We consider the decoding of a set of NC packets, and allow temporary buffering for future decoding opportunities. Our use of linear network coding therefore leads to a more flexible and efficient anchor recovery.

The work in [27] addresses mitigating burst loss by multi-path multiple description coding (MDC). Ren *et al.* study the problem of allocating anchors at users for a peer-to-peer wired network, with the objective to minimize the streaming cost [28]. Jacob studies transmission policy for point-to-point multiview content transmission over bandwidth-limited channels, for both MVC and texture-plus-depth coding [29]. In contrast to them, we study here the joint problem of anchor pulling and NC generation for a broadcast-based wireless live FVV streaming.

Our cooperative wireless live FVV streaming network is orthogonal and complementary to cellular network broadcasting such as Multimedia Broadcast Multicast Service (MBMS) [30]. In MBMS, anchors are pushed (broadcast) to users irrespective of user locations and demands. Due to its limited channel bandwidth, it is best deployed for popular videos of wide geographical interest. PAFV, as a pull-based algorithm, is most suitable and cost-effective for a live stream of local significance (e.g., a video of common interest to a local user group). It also achieves better video quality and bandwidth utilization through anchor pulling and NC broadcasting.

## III. Middleware Design and Problem Formulation

We first describe in Section III-A the proposed middleware for the collaborative wireless live FVV streaming network. We then formulate the distortion minimization problem in Section III-B.

### A. Middleware Framework

In live FVV streaming, each anchor packet has a playback deadline of $T$ (seconds), i.e., the maximum tolerable playback delay. (Time synchronization between clients and the server can be achieved by protocols such as network time protocol (NTP) or precision time protocol (PTP).) Before the playback deadline of a packet, each node can cache it and share it with neighbors to increase the packet availability.[4]

The time is divided into slots. We show in Fig. 3 a slot-based timeline for a client in the system. The duration of each slot is $\Delta$, where $\Delta \leq T$. During each slot, each client performs the following three operations:

[4]Note that such a buffering requirement is little, given the memory capacity of today's mobile device (e.g., considering an FVV anchor stream of about 500 kb/s and a deadline of 2 seconds, such buffering requirement is only 1 Mbit).

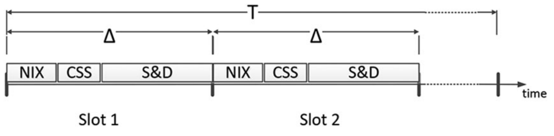


Fig. 3. Slot-based operation.

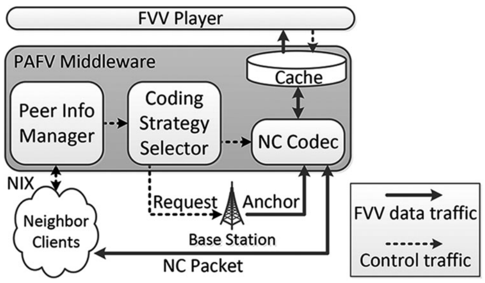


Fig. 4. Proposed middleware for wireless live FVV streaming with NC-based sharing.

1) *neighbor information exchange (NIX):* each node broadcasts to neighbors its information such as available anchor packets and cached NC packets, in terms of an FVV packet buffermap;
2) *coding strategy selection (CSS):* each node, according to the neighbor information and its own available bandwidth, jointly determines what anchors to pull from the server and an NC strategy; and
3) *sharing and decoding (S&D):* each node, according to the computed NC strategy, codes and broadcasts NC packets to its neighbors. Each node tries to decode some received NC packets, and caches the rest for decoding opportunities in the future.

The above operations are repeated for packets of each playback time instance throughout the live streaming session. In this slot-based system, topology dynamics (due to node mobility, join/leave etc.) can be timely reflected by the neighbor information exchange in each slot. *CSS* therefore computes a new best NC strategy based on the most up-to-date neighbor information within the slot.

In a relatively static network with low channel loss rates, the collective NC strategy would mostly remain stable over time, leading to reduced anchor-request traffic. As CSS decisions in different slots are independent, we can then focus on an arbitrary slot of a client and optimize NC strategy to minimize video distortion by the end of that time slot. We show in Fig. 4 the proposed middleware running at each client. The middleware connects to server via a primary channel, and to neighbor clients via a secondary channel. The middleware also serves application layer FVV player. Modules and their functions of the middleware are summarized in Table I.

In Fig. 4, the flows of FVV data and control traffic are shown in solid and dashed lines, respectively. *Peer Info Manager* receives and keeps track of neighbor information, mainly the availability of anchors and any buffered NC packet in neighbor nodes, as well as the available channel bandwidths.

TABLE I
MODULES OF THE MIDDLEWARE

| Module | Description |
|---|---|
| Peer Info Manager | Monitors neighbor information. |
| Coding Strategy Selector | Computes the NC strategy. Request new anchors if needed (via primary channel). |
| NC Codec | Performs NC coding/decoding. Sharing coded NC packets (via secondary channel). |
| Cache | Stores anchors packets and NC packets. |

TABLE II
MAJOR SYMBOLS USED IN THE PAPER

| Symbol | Description |
|---|---|
| $\mathcal{N}$ | The set of all nodes. |
| $\mathcal{N}_i$ | The set of neighbor nodes of $i$, including $i$. $\mathcal{N}_i \subseteq \mathcal{N}$. |
| $\mathcal{U}$ | The set of all the views. |
| $\hat{\mathcal{A}}$ | The set of all anchor views. $\hat{\mathcal{A}} \subseteq \mathcal{U}$ |
| $\mathcal{A}_i$ | The set of anchors available at node $i$ when the slot begins. $\mathcal{A}_i \subseteq \hat{\mathcal{A}}$. |
| $\mathcal{A}'_i$ | The set of anchors available at node $i$ when the slot ends. $\mathcal{A}'_i \subseteq \hat{\mathcal{A}}$. |
| $\mathcal{S}_i$ | The set of NC packets received by node $i$. |
| $c$ | Anchor streaming rate (bps). |
| $p$ | Anchor packet size (bits). |
| $r_i$ | Primary channel bandwidth of node $i$ (bps). |
| $r'_i$ | Secondary channel bandwidth of node $i$ (bps). |
| $T_{\mathrm{sd}}$ | *S&D* duration (seconds). |
| $I_i^a$ | Optimizing parameter, equals to 1 if node $i$ pulls anchor packet $a$ from the server, and 0 otherwise. |
| $B_i^{a,s}$ | Optimizing parameter, equals to 1 if node $i$ selects anchor packet $a$ to code NC packet $s$, and 0 otherwise. |
| $K_i$ | Optimizing parameter, equals to 1 if node $i$ broadcasts its NC packets, and 0 otherwise. |
| $D(u, \mathcal{A})$ | Minimum achievable distortion of rendered view $u$, given anchor set $\mathcal{A}$ (dB). |
| $\pi_u$ | Popularity of view $u$. |

When a *CSS* operation is triggered, *Peer Info Manager* feeds neighbor information into *Coding Strategy Selector*, which jointly determines the NC strategy and anchors needed to be pulled from the server. *CSS* can easily adapt to cases where anchors pulled at each node are predetermined (i.e., pushed by the server) by optimizing the NC strategy with only the received anchors.

During *S&D*, the NC strategy is received by *NC Codec*, which retrieves anchors from *Cache* and performs network coding to generate corresponding NC packets. *NC Codec* also decodes each received NC packet if decodable, and then stores either the decoded anchors or the NC packet in *Cache*. When last slot is ended, anchors stored in *Cache* are then used by *FVV Player* for rendering and playback.

## *B. Decision Variables and Problem Objective*

We focus on the CSS operation at each node in an arbitrary slot. Each anchor therefore is represented by a single source packet. The major symbols used in our problem formulation are listed in Table II. Let $\mathcal{U}$ and $\hat{\mathcal{A}}$ be the set of all the views and the set of all the anchors, respectively.[5]

[5]We consider the fine-grained discrete view set, though our work can be easily extended to continuous view space.

Let $\mathcal{A}_i$ be the set of anchors available at node $i$ when the slot begins. Let $\mathcal{N}_i$ be the set of neighbors of $i$. With NIX exchange, node $i$ learns $\mathcal{A}_j$ for every $j \in \mathcal{N}_i$. $i$ then needs to make the following two decisions in CSS operation. Denote $I_i^a \in \{0,1\}$ as the binary variable indicating whether node $i$ pulls anchor $a \in \hat{\mathcal{A}}$ from the server. Denote $B_i^{a,s} \in \{0,1\}$ as the binary variable indicating whether $i$ uses $a$ to code its NC packet $s$. Further denote $K_i \in \{0,1\}$ as the binary variable indicating whether $i$ broadcasts its NC packets. Node $i$ with $K_i = 1$ then shares $s$ with its neighbors. Each node accumulates a set of received NC packets at the end of the slot, decodes and recovers some anchors, and forms a new and larger anchor set, denoted as $\mathcal{A}_i'$ for node $i$.

Denote $D(u, \mathcal{A})$ as the distortion function to evaluate the optimal distortion for rendering view $u$ provided anchor set $\mathcal{A}$. Further denote $\pi_u$ as the given spatial popularity of view $u$. Our objective is to minimize the *overall expected distortion* for every view at every node, i.e., to find

$$\min_{I_i^a, B_i^{a,s}, K_i} \sum_{i \in \mathcal{N}} \sum_{u \in \mathcal{U}} \pi_u D(u, \mathcal{A}_i') \tag{1}$$

where the overall expected distortion is computed by first weighing the distortion of a rendered view at a node by its popularity, then taking the summation over all the views and all the nodes. Note that our problem and formulation is general enough independent of the specific forms of $D$ and $\pi$.

We now model $A_i'$ at node $i$. Suppose node $i$ accumulates an NC packet set $\mathcal{S}$, where the subset $\mathcal{R} \subseteq \mathcal{S}$ is the decodable NC packet set of the *maximum cardinality*. In other words, $\mathcal{R}$ is the largest subset of $\mathcal{S}$ such that the anchors used to code $\mathcal{R}$ *and* are missing at $i$ is no more than $|\mathcal{R}|$, in order for a full-rank coefficient matrix to be formed, i.e.

$$\mathcal{R} = \arg\max_{s: s \subseteq \mathcal{S}} |s| \text{ s.t.} | \mathcal{A}^{\mathcal{R}} - \mathcal{A}_i \bigcup \{a : a \in \hat{\mathcal{A}}, I_i^a = 1\}| \leq |\mathcal{R}| \tag{2}$$

where $\mathcal{A}^{\mathcal{R}}$ is the anchor set used to code $\mathcal{R}$, i.e.

$$\mathcal{A}^{\mathcal{R}} \triangleq \left\{ a : a \in \hat{\mathcal{A}}, \sum_{j \in \mathcal{N}_i} \sum_{s \in \mathcal{R}} K_j B_j^{a,s} \geq 1 \right\}. \tag{3}$$

Consequently, $A_i'$ at node $i$ after the decoding of $\mathcal{R}$ can be written as

$$\mathcal{A}_i' \triangleq \mathcal{A}_i \bigcup \mathcal{A}^{\mathcal{R}} \bigcup \{a : a \in \hat{\mathcal{A}}, I_i^a = 1\}. \tag{4}$$

## C. Constraints

We discuss each of the constraints of our problem in this section.

Node $i$ can only use anchors either in $A_i$ or pulled to code NC packet. We therefore require

$$B_i^{a,s} \leq 1_{a \in \mathcal{A}_i} + I_i^a \tag{5}$$

where $1_{a \in \mathcal{A}_i} = 1$ if $a \in \mathcal{A}_i$, and 0 otherwise.

For the primary channel there is a bandwidth constraint. The sum of the transmission rates of all the anchors that node $i$ simultaneously pulls should not exceed its primary channel bandwidth $r_i$, i.e.

$$c \sum_{a \in \hat{\mathcal{A}}} I_i^a \leq r_i \qquad \forall i \in \mathcal{N} \tag{6}$$

where $c$ is the anchor streaming rate.

For the secondary channel sharing, we have two constraints. The first one is the given *S&D* duration length $T_{\text{sd}}$. NC packet sharing duration at any node should not exceed $T_{\text{sd}}$. Denote $r_i'$ as the secondary channel bandwidth at node $i$, and $p_s$ as the size of NC packet $s$. Further denote $S$ as the set of NC packets coded by $i$, we hence have

$$\sum_{s \in S} \frac{p_s}{r_i'} \leq T_{\text{sd}} \qquad \forall i \in \mathcal{N} \tag{7}$$

where $p_s$ consists of the size of a single anchor and a small $\epsilon$ for NC coefficients, i.e.

$$p_s = p + \epsilon \tag{8}$$

in which $p$ denotes the anchor size. $\epsilon$ depends on the overall number of anchors and the number of anchors coded in $s$, and is normally small comparing to the payload.

The second constraint of the secondary channel is the bandwidth constraint, i.e., the transmission rate of any NC packet must be lower than that of the channel in order to preserve the video coding rate of each anchor. Packet transmission rate equals packet size divided by transmission time. NC packet $s$ must be transmitted within the time used to transmit an anchor to retain the video coding rate. Let $t_s$ be the transmission time of $s$, we therefore have

$$t_s = \frac{p_s}{r_i'}. \tag{9}$$

With NC packet size defined by (8) and transmission time requirement defined by (9), we can compute the required coding rate of $s$, denoted as $c_s$, by

$$c_s = \frac{p_s}{t_s}. \tag{10}$$

We then state the secondary channel bandwidth constraint as

$$c_s \leq r_i' \qquad \forall i \in \mathcal{N}. \tag{11}$$

For collision avoidance, we require no more than one neighbor of each node can broadcast its NC packets, i.e.

$$\sum_{i \in \mathcal{N}_j} K_i \leq 1 \qquad \forall j \in \mathcal{N}. \tag{12}$$

By definition, we also require

$$I_i^a, B_i^{a,s}, \ K_i \in \{0,1\} \qquad \forall i \in \mathcal{N}, a \in \hat{\mathcal{A}}. \tag{13}$$

Our problem can then be stated as follows: determine $I_i^a$, $B_i^{a,s}$ and $K_i$ $\forall i \in \mathcal{N}, a \in \hat{\mathcal{A}}$ for the Objective (1), subject to Constraints (5) –(7) and (11) –(13).

The problem is NP-hard, as shown in the Appendix.

## IV. PAFV: Distributed and Collaborative FVV Streaming

In this section we propose PAFV (Peer-Assisted Freeview Video), a novel and fully distributed algorithm for collaborative FVV streaming. PAFV is designed to operate in each individual time slot as shown in Fig. 3. Within each time slot, a node performs three operations, NIX, CSS and S&D, discussed in Sections IV-A, IV-B and IV-C, respectively. The time complexity is analyzed in Section IV-D.

### A. Neighbor Information Exchange (NIX)

Each node shares its information with its neighbors by broadcasting a *beacon* message. Such beacon packet is of small size, and hence has high delivery ratio (note that PAFV also does not require perfect beacon delivery to work). This control packet includes an FVV buffermap that indicates locally available anchors. The buffermap can be efficiently constructed using a bit array of size $[|\hat{A}|]$, with each field corresponding to an anchor. Each field is set to 1 if the corresponding anchor is locally available, and 0 otherwise. Information about locally cached NC packets, such as the IDs of anchors coded in each NC packet, is also included. With such information, a node is able to predict for each candidate NC packet, the decodability of both the candidate and cached NC packets at each neighbor. The node can then derive the available anchors and subsequently the expected distortion reduction at each neighbor.

### B. Coding Strategy Selection (CSS)

Based on the received neighbor information, each node then tries to determine its own NC strategy. PAFV adapts a heuristic algorithm that aggressively reduces overall expected distortion in generating each NC packet.

Suppose node $i$ shares an NC packet $s$ to its neighbors. With the knowledge of both anchors and buffered NC packets at neighbor $j, i$ can derive $\mathcal{A}_j^s$, the set of available anchors at $j$ after decoding $s$, for each $j \in \mathcal{N}_i$, according to (4).

We now define the *unit gain* of NC packet $s$ to evaluate its efficiency. The unit gain of $s$ describes the amount of overall expected distortion reduction brought by $s$ for every bit transmitted (i.e., dB/bit). Formally, the unit gain of NC packet $s$ is defined as

$$g_i^s \triangleq \frac{\sum_{j\in\mathcal{N}_i}\sum_{u\in\mathcal{U}}\pi_u\left(D(u,\mathcal{A}_j)-D(u,\mathcal{A}_j^s)\right)}{p_s+\sum_{a\notin\mathcal{A}_i} I_i^a p_a} \tag{14}$$

where the numerator sums the total expected distortion reduction of all the views $u \in \mathcal{U}$ at every neighbor $j \in \mathcal{N}_i$, given the new anchor set $\mathcal{A}_j^s$. The denominator sums both the NC packet size $p_s$, and the total primary channel pulling traffic required. Note that $\mathcal{N}_i$ includes node $i$, so unit gain considers the distortion reduction at $i$ itself.

With unit gain of an NC packet defined, node $i$ now codes the NC packet set $\mathcal{S}$ to solve the problem

$$\max_{I_i^a, B_i^{a,s}} \sum_{s\in\mathcal{S}} g_i^s \tag{15}$$

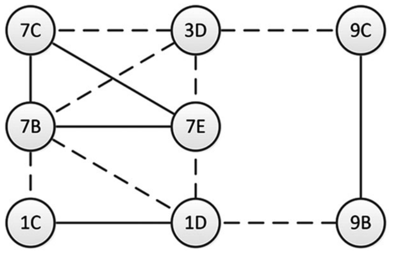


Fig. 5. Graph constructed at node $A$ of Fig. 2 before sharing. Node ID $xy$ corresponds to packet $x$ missing at node $y$.

subject to primary channel bandwidth constraint [corresponding to Constraint (6)]

$$c\sum_{a\notin\mathcal{A}} I_i^a \le r_i \tag{16}$$

*S&D* duration constraint [corresponding to Constraint (7)]

$$\sum_{s\in\mathcal{S}} \frac{p_s}{r_i'} \le T_{\mathrm{sd}}, \tag{17}$$

and secondary channel bandwidth constraint [corresponding to Constraint (11)]

$$c_s \le r_i', \quad \forall s \in \mathcal{S} \tag{18}$$

where $p_s$ and $c_s$ are calculated according to (8) and (10), respectively.

We adapt the clique-finding approach [25], [31] to address the problem as follows.

Given a set of neighbors with their missing and available anchors, a graph $G$ can be efficiently constructed by introducing a vertex $a_{\mathrm{mj}}$ for each missing anchor $m$ at each node $j$. Two vertices $a_{m_1j_1}$ and $a_{m_2j_2}$ are connected if either one of the following holds:

1) *Rule 1:* $m_1 = m_2$; or
2) *Rule 2:* $a_{m_1}$ is available at $j_2$, and $a_{m_2}$ is available at $j_1$.

To avoid finding infeasible cliques, we further require the following two rules for each connection:

3) *Rule 3:* the total required transmission rate of $a_{m_1j_1}$ and $a_{m_2j_2}$ must be within the primary channel bandwidth limit; and
4) *Rule 4:* the required transmission rate of $a_{m_1j_1}$ and $a_{m_2j_2}$, if coded together, must be within the secondary channel bandwidth limit.

If $a_{m_1j_1}$ and $a_{m_2j_2}$ are connected, a single NC packet coded with $a_{m_1j_1}$ and $a_{m_2j_2}$ can then recover both missing packets at $j_1$ and $j_2$. As a result, all missing packets occurred in a clique (i.e., a complete subgraph) of $G$ can be recovered by a single NC packet. We can then solve (15) by constructing $G$ at node $i$ for its neighbors $\mathcal{N}_i$, and finding a set of cliques that collectively leads to maximal unit gain.

The graph $G$ constructed at node $A$ of Fig. 2 is shown in Fig. 5. Node ID $7C$ suggests that anchor 7 is missing at node $C$, likewise for others. Solid and dotted edges correspond to *Rules 1* and *2*, respectively. We do not consider *Rules 3* and *4* in this example. The largest clique is obviously formed by the set $\{7C, 7B, 3D, 7E\}$, leading to an NC packet coded by anchor set $\{3, 7\}$ to recover these 4 missing packets. For the second NC packet, there are multiple choices, as cliques formed by $\{1C, 1D\}, \{9B, 9C\}$ and $\{1D, 9B\}$ all have the same unit gain. So any one of the NC packets coded by $\{1\}, \{9\}$ or $\{1, 9\}$ may be chosen and transmitted to recover 2 corresponding missing packets.

Maximal clique enumeration is known to be NP-hard [32]. We therefore propose *greedyCSS* Algorithm, summarized in Algorithm A, to tackle the problem efficiently. Function *sort()* sorts the input list according to the unit gain; *buildG()* constructs the graph from neighbor information according to the four rules; *findClique()* returns the clique containing a given packet, such that the corresponding NC packet has the highest unit gain with buffered NC packets considered.

We next describe the procedure of *greedyCSS* operated at node $i$ in detail. First, all missing packets in the neighborhood of node $i$ are sorted in descending order according to their unit gain (Line 5). Note that the sorted list $M$ may contain multiple references of the same anchor $a$, if it is missing at multiple neighbors.

Next, the graph $G$ is constructed according to neighbor information, i.e., missing packets and available packets at each neighbor (Line 6).

Next we generate multiple cliques as candidates. While not all missing packets have been dealt with (Lines 8 to 15), the missing packet $a$ with the highest unit gain is chosen, since it is logical to prioritize such anchor. PAFV then locates all occurrences of $a$ in $G$, and looks for the maximal clique containing $a$. (Line 9).

To find the desired clique ($clq$ in Alg. A) containing $a$, *findClique()* employs Feige's approximation algorithm [33].[6] The search space is first reduced to a subgraph containing all appearances of $a$ and their neighbors. The subgraph is then fed into Feige's algorithm to look for cliques each containing at least one appearance of $a$. Each resulting clique corresponds to an NC packet. If multiple cliques are found, *findClique()* returns the one with the highest unit gain. The algorithm is modified to take into consideration of any buffered NC packet at each neighbor when computing the unit gain of a clique.

The clique returned by *findClique()* is then removed from $G$ (Line 11). All the missing packets covered by this clique are also removed from $M$ (Lines 13).

Note that the clique containing the highest-unit-gain anchor may not have an overall highest unit gain. A sorted list $s\_temp$ is therefore maintained to store all the cliques found so far, in the order of their unit gains. As cliques are found one by one, and can be efficiently inserted in $s\_temp$ by function *insert()*, there is no need to perform separate sorting. After all missing packets are dealt with, we have the sorted candidate NC packets stored in $s\_temp$.

Finally, a top NC packet set is selected and returned as the computed NC strategy (Lines 24).

In summary, *CSS* first selects the anchor missing by at least one neighbor with the highest unit gain, then finds candidate NC packets containing that anchor (using clique-finding), and selects the one with the highest unit gain as a candidate.*CSS* processes each anchor to generate multiple NC packet candidates. The candidate set with the highest aggregated unit gain are then selected as the returned NC strategy.

[6]We do not focus on designing a novel clique-finding algorithm, hence any other algorithm may also be adapted.

**Algorithm A:** *greedyCSS.*

```
1: INPUT: $\mathcal{A}_j \forall j \in \mathcal{N}_i, r_i, r'_i, T_{\text{sd}}$
2: OUTPUT: $s$ (NC packet set to be coded)
3: $s \leftarrow \phi$
4: $s\_temp \leftarrow \phi$
5: $M \leftarrow \text{sort}(\{a_j : a_j \in (\hat{\mathcal{A}} - \mathcal{A}_j), \forall j \in \mathcal{N}_i\})$
6: $G \leftarrow \text{buildG}(\{\mathcal{A}_j\})$
7: while $M \neq \phi$ do
8:         $a \leftarrow \text{head}(M)$
9:         $clq \leftarrow \text{findClique}(G, a, r_i, r'_i)$
10:        $s\_temp \leftarrow s\_temp.\text{insert}(clq)$
11:        $G \leftarrow G.\text{rm}(clq)$
12:        for $p \in clq$ do
13:                $M \leftarrow M - p$
14:        end for
15: end while
16: $t \leftarrow T_{\text{sd}}$
17: while $t \geq 0$ and $r_i \geq 0$ AND $s\_temp \neq \phi$ do
18:        $nc \leftarrow \text{head}(s\_temp)$
19:        $s.\text{add}(nc)$
20:        $\text{pop}(s\_temp)$
21:        $t \leftarrow t - \text{getNCSize(nc)}/r'_i$
22:        $r_i \leftarrow r_i - \sum_{p \in nc, p \notin A_i} c$
23: end while
24: return $s$
```

## C. NC Broadcasting and Decoding

The NC packet set $\mathcal{S}$ coded at node $i$ are then broadcasted to its neighbors, who receive and decode any decodable NC packet set received. We prefer nodes with high unit gain of NC packet set to broadcast, so node $i$ waits a small broadcast delay $t$ inversely proportional to the unit gain for scheduling, i.e.

$$t \propto \frac{1}{\sum_{s \in \mathcal{S}} g_i^s}. \tag{19}$$

During this delay, if NC broadcasting from a neighbor is received, $i$ then suppresses its broadcasting in this slot.

Instead of waiting until the end of sharing period, a node can start decoding progressively with Gauss-Jordan elimination as NC packets arrive. NC packets not yet decodable are cached for future decoding possibilities. At the end of the playback deadline, undecodable NC packets are dropped, and decoded anchors are used for virtual view rendering and playback.

## D. Complexity Analysis

In this section we briefly analyze the time complexity of PAFV.

For *NIX* (Section IV-A), the major operation is generating the buffermap. However, the buffermap can be updated along with each incoming data packet, so this step takes constant time.

For *CSS* (Section IV-B), we need to first compute and sort the unit gain of each missing anchor in the neighborhood. There are at most $|\hat{\mathcal{A}}|$ distinct missing packets, and calculating the unit gain of each one takes $O(|\mathcal{U}|)$. Identifying any decodable NC packets cached by neighbors can take $O(1)$ if a hash table is used. Overall time complexity of computing the unit gains is therefore $O(|\hat{\mathcal{A}}||\mathcal{U}|)$. Sorting them takes $O(|\hat{\mathcal{A}}|\log|\hat{\mathcal{A}}|)$.

The construction of the graph takes $O(|\mathcal{N}|^2|\hat{\mathcal{A}}|)$ according to [31]. Given a graph containing a clique of size $n/k$, and a parameter $t \ll n/k$, Feige's max clique algorithm takes $O(\binom{t}{2t}n^{c'})$ for some constant $c' < 1$ [33].[7] By setting $t = \Theta(\log n/\log\log n)$, $\binom{2t}{t}$ becomes polynomial [33], and turned out to be $O(n^2)$ if $k = 1$. $O(\binom{2t}{t}n^{c'})$ can then be reduced to $O(n^{c'+2})$, which is upper-bounded by $O(n^3)$. Finding a maximum clique containing an appearance of $a$ is equivalent to finding a maximum clique in the subgraph induced by the neighbors of that appearance of $a$. The number of appearance of $a$ is at most $|\mathcal{N}|$, so is the number of such subgraphs, each containing at most $|\hat{\mathcal{A}}||\mathcal{N}|$ nodes (if every packet is missing at every neighbor). Therefore, finding maximal cliques in all these subgraphs takes $O(|\mathcal{N}|(|\hat{\mathcal{A}}||\mathcal{N}|)^3)$. Computing the unit gain of each clique alongside takes $O(|\mathcal{U}|^2)$. Inserting a newly found clique to the sorted list using binary search takes a logarithmic time with respect to the number of cliques in the list. The overall complexity of *findClique()* is hence $O(|\mathcal{N}|(|\hat{\mathcal{A}}||\mathcal{N}|)^3|\mathcal{U}|^2)$, i.e., $O(|\mathcal{N}|^4|\hat{\mathcal{A}}|^3|\mathcal{U}|^2)$. Clique removal and covered packet removal both take constant time.

The aggregate time complexity of Algorithm *greedyCSS* is $O(|\hat{\mathcal{A}}||\mathcal{U}| + |\hat{\mathcal{A}}|\log|\hat{\mathcal{A}}| + |\mathcal{N}|^2|\hat{\mathcal{A}}| + |\mathcal{N}|^4|\hat{\mathcal{A}}|^3|\mathcal{U}|^2)$, which is $O(|\mathcal{N}|^4|\hat{\mathcal{A}}|^3|\mathcal{U}|^2)$, i.e., dominated by clique finding.

For $S$ & $D$ (Section IV-C), the encoding of NC packets is linear in terms of anchors used, i.e., $O(|\hat{\mathcal{A}}|)$, while the decoding takes $O(|\hat{\mathcal{A}}|^3)$ with Gauss-Jordan elimination.

As a result, the overall slot operation complexity is $O(|\mathcal{N}|^4|\hat{\mathcal{A}}|^3|\mathcal{U}|^2) + O(|\hat{\mathcal{A}}|) + O(|\hat{\mathcal{A}}|^3)$, which is $O(|\mathcal{N}|^4|\mathcal{U}|^5)$ if we relax $O(|\hat{\mathcal{A}}|)$ to be $O(|\mathcal{U}|)$.[8]

## V. Illustrative Simulation Results

We have conducted extensive simulation study on PAFV. We first present the simulation setup in Section V-A, followed by illustrative results in Section V-B.

### A. Simulation Setup

In our numerical simulation, nodes are randomly and uniformly located over a 100 m × 100 m area, with a node transmission range of 25 m. Node mobility follows the random waypoint model [34]. Primary and secondary channel bandwidths both follow truncated normal distribution, with means 7 units and 10 units, respectively. Link loss rate for anchor/NC packets is independent and identically distributed (i.i.d.) with mean 2%. Due to its small size, beacon loss is assumed to be negligible. Anchor packet size is 5 units. There are in total 11 views, with 6 of them as anchors. The playback delay $T$ is 3 slots, each with a total duration of 3 seconds, and an $S\&D$ duration $T_{\rm sd} = 2$ s. Unless otherwise stated, we use the baseline parameters summarized in Table III.

TABLE III
BASELINE PARAMETERS

| Parameters | Value |
|---|---|
| Area | $100$ m $\times$ $100$ m |
| Tx range | $25$ m |
| Link loss | i.i.d.,$(\mu,\sigma) = (2\%, 0.1\%)$ |
| Number of nodes | $12$ |
| Mobility | $v_{\rm max} = 5$ m/s, $T_{\rm pause} = 3$ s |
| Primary bandwidth | $r_i \sim \mathbb{N}(7,3)$ units/s, left truncated |
| Secondary bandwidth | $r_i' \sim \mathbb{N}(10,3)$ units/s, left truncated |
| Total number of views | $|\mathcal{U}| = 11$ |
| Anchors | $\hat{\mathcal{A}} = \{1,3,5,7,9,11\}$. |
| Anchor packet size | $p_a = 5$ units |
| View popularity | $\alpha = \beta = 0.6$ in Eq. (21) |
| Optimal view PSNR | 40 dB |
| Playback delay | $T = 3$ slots |
| *S&D* duration | $T_{\rm sd} = 2$ s |

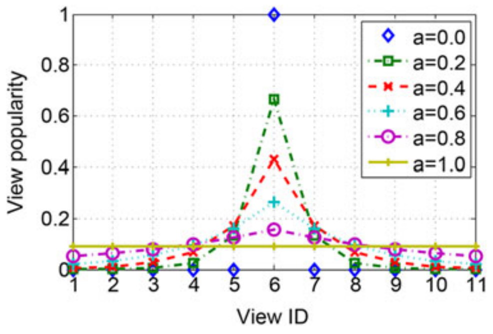


Fig. 6. View popularities generated using (21).

PAFV formulation does not assume any particular popularity distribution. We use the following popularity for illustration purpose. Let $j$ be the most popular view, we generate view popularity $\forall 1 \le i \le |\mathcal{U}|$ by

$$p_i = \begin{cases} \alpha^{j-i}, & i < j \\ 1, & i = j \\ \beta^{i-j}, & i > j \end{cases} \tag{20}$$

where $\alpha \in [0,1]$ and $\beta \in [0,1]$ characterize the distribution. We then normalize each $p_i$ by

$$p_i' = \frac{p_i}{\sum_{i=1}^{|\mathcal{U}|} p_i} \tag{21}$$

and use $p_i'$ as the popularity of view $i$. Fig. 6 shows the generated view popularities with $j = 6$, and varying values of $\alpha$ $(\beta = \alpha)$. Note that when $\alpha = 0$, only view $j$ has a non-zero popularity with value one; as $\alpha$ increases, the popularity distribution becomes more uniform.

The study in [4] states that the distortion of a synthesized view is a quadratic function of the spatial distances between the view and its left and right nearest available anchors, and becomes cubic when the distances are large. We adapt the function and extend it as we allow view rendering from anchors of only one

[7] $n^{c'}$ is bounded by the number of phases in Feige's algorithm, which is less than $n - n/2k$.

[8] In practice, the size of $\hat{\mathcal{A}}$ is often much smaller than that of $\mathcal{U}$.

TABLE IV
COMPARISON SCHEME CATEGORIZATION

| | Cooperative | Non-cooperative |
|---|---|---|
| With NC | PAFV | N/A |
| W/o NC | GAS | RAS |

side. Denote $x$ and $y$ are the distances between the synthesized view and its left and right available anchors respectively. If anchors of only one side is available, we use $x$ to denote the shortest distance between the view and an anchor. Our distortion function $D_A^u$ for virtual view $u$ given anchor set $A$ is therefore

$$D_A^u \triangleq \begin{cases} a_1x^2 + a_2y^2 + a_3x + a_4y + a_5, & \text{if } u \text{ is sandwiched} \\ a_6x^3 + a_7, & \text{otherwise.} \end{cases} \tag{22}$$

We find the function constants by polynomial surface fitting over the rendering result set of the multiview sequence "Kendo" [35], and use the function in our subsequent simulations. Note that our formulation and PAFV algorithm do not rely on any specific distortion function, nor is optimizing the function our study focus. Other functions would also work as long as they appropriately relate the distortion of a rendered view to available anchors.

We compare PAFV with the following two approaches:

1) *greedy anchor selection (GAS)*, in which each node pulls from server the anchors that lead to maximal distortion reduction in the neighborhood, and then shares top ones with its neighbors; and
2) *random anchor selection (RAS)*, in which each node uniformly randomly selects anchors to pull from server, and shares a random subset with their neighbors.

These two approaches, together with PAFV, can be categorized as shown in Table IV.

In the simulation we evaluate the following metrics of different algorithms. 1) *Overall expected distortion*, computed according to (1), i.e., by first weighing the distortion of a rendered view at a node by its popularity, then taking the summation of this weighted distortion over all the views at all the nodes. 2) *Network redundancy*, which is the average number of duplicates each anchor experienced at a node. It reflects the transmission efficiency in distortion reduction.

## B. Illustrative Results

We plot the overall expected distortion versus network size in Fig. 7(a). The distortion drops in all three schemes as the network becomes dense, due to the increased anchor availability in a neighborhood. PAFV reduces the overall network distortion the fastest. This is because each neighborhood now contains more nodes for cooperation, increasing the anchor availability at each node. GAS, also targets at maximum distortion reduction, enjoys similar but slower distortion decrease, as simple forwarding without NC is less efficient. For RAS, neighboring nodes may pull overlapped anchor sets, leading to high duplication and hence low anchor diversity. The distortion distribution among the nodes is presented in Fig. 7(b). Most nodes experience medium-to-low distortion in PAFV, indicating a good fairness. In all three algorithms, there are nodes with insufficient primary channel bandwidth, and are solely rely on the neighbors. They therefore often experience relatively high distortion.

We plot the overall expected distortion under different network conditions in Fig. 8. We show in Fig. 8(a) the overall network distortion as the mean primary channel bandwidth increases. The overall distortions of all the algorithms decrease as more anchors can be injected into the network by the server. RAS suffers from the highest distortion due to high duplication. GAS, although enjoys a lower distortion due to neighbor cooperation, still performs worse than PAFV when primary bandwidth is limited. When primary channel bandwidth becomes sufficient to forward most of the popular anchors without NC, the effectiveness of NC gradually reduces. PAFV outperforms RAS by cooperative sharing, and outperforms GAS due to the effectiveness of NC, and converges with GAS when primary channel bandwidth is sufficiently large.

Next we evaluate PAFV under different primary channel bandwidth distributions. To facilitate this experiment, we randomly and uniformly reassign the primary channel bandwidth of some nodes to others. Note that a node is only a *puller* if its primary channel bandwidth is sufficient for at least one anchor. The result shown in Fig. 8(b) suggests that PAFV performs best when the primary channel bandwidth is more distributed. This is because a reduced hop distance from any node to its nearest puller leads to more effective time slots for cooperative sharing. In the case where the network relies on a few pullers, an increased overall FVV distortion is observed, due to the fact that nodes far away from a puller waste many time slots waiting for the anchors to arrive. In the worst case some nodes cannot be served if their hop distances to the nearest pullers exceed the playback deadline allowed, and are left suffering from the maximum video distortion.

We show in Fig. 8(c) how PAFV performs under different link loss conditions. All three schemes suffer from higher distortion when link loss rate increases. Shared packets, optimized in PAFV and GAS, are of different distortion reduction effectiveness. Therefore, PAFV and GAS are more sensitive to the loss of important packets. The distortion growth of RAS on the other hand, is relatively insensitive due to the randomness in pulling and sharing. With the optimization in packet sharing, both PAFV and GAS outperform RAS in low-to-medium link loss. With the effectiveness of NC, the distortion growth of PAFV is significantly slower than that of GAS.

We show in Fig. 9 the overall FVV distortion with different playback delays. In all three algorithms, overall distortions decrease with more time slots available. PAFV enjoys the lowest overall distortion thanks to the effectiveness of cooperation and the efficiency of NC. We notice that the distortion drops sharply at first, and gradually slows down. This is because at first many anchors are missing, so the recovery of any subset of those anchors may leads to a drastic distortion reduction. Later when only a few anchors are still missing, together with possible duplications in sharing, the distortion reduction slows down.

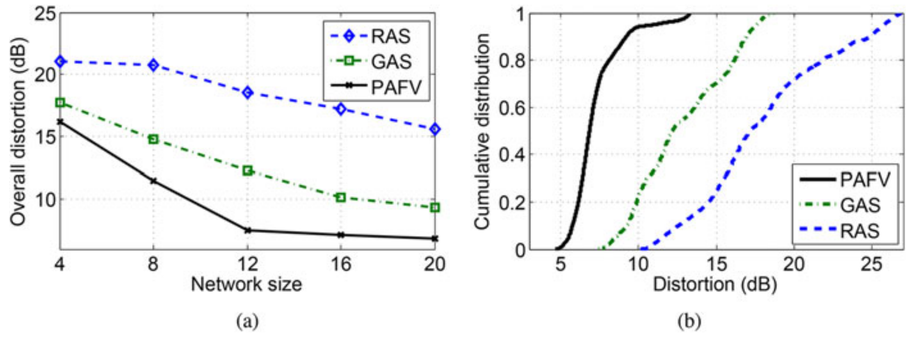


Fig. 7. Overall expected distortion versus network size. (a) Overall distortion versus number of nodes. (b) Cumulative distribution of overall distortion.

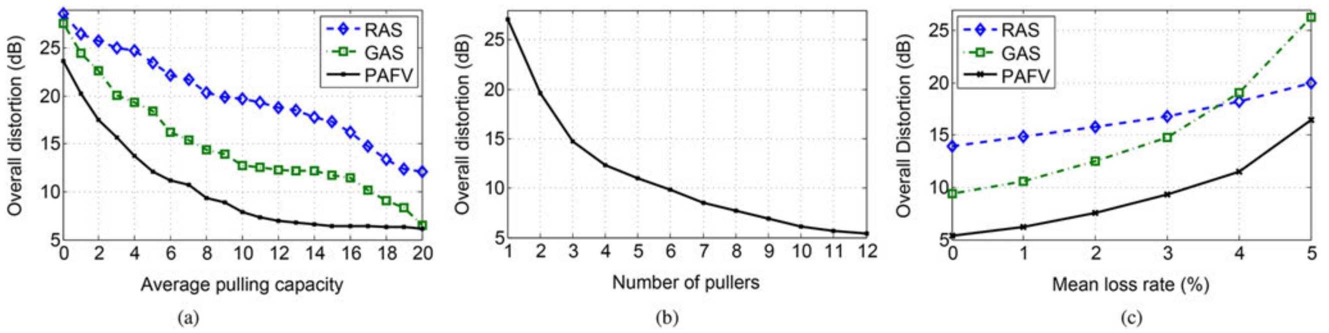


Fig. 8. Overall expected distortion in different network conditions. (a) Overall distortion versus average primary channel bandwidth. (b) Overall distortion versus number of pullers. (c) Overall distortion versus link loss rate.

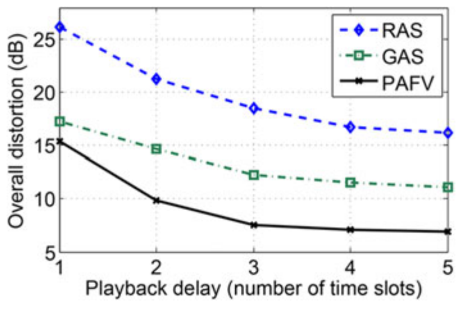


Fig. 9. Overall distortion versus playback delay.

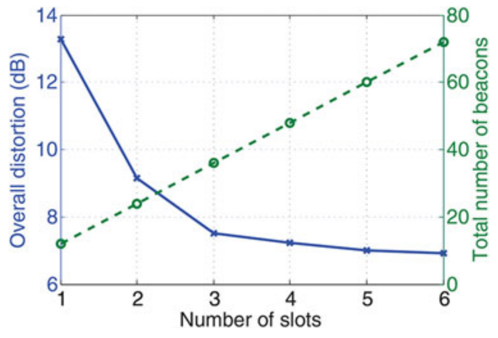


Fig. 11. Trade-off between number of slots and slot duration.

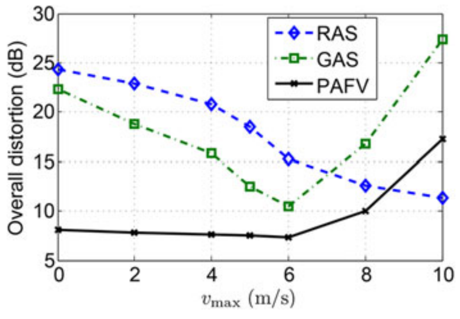


Fig. 10. Distortion versus $v_{\max}$.

We show in Fig. 10 the effect of node mobility (in terms of $v_{\max}$). All three algorithms enjoy slight distortion drop when the maximum moving speed increases but remains low. When moving speed becomes high, the optimization decision made by PAFV and GAS becomes less effective as the neighbor set changes too fast, leading to a distortion increase. RAS performs worst when the network is relatively static due to high degree of redundancy, but improves as nodes become more mobile, due to more chances for nodes to exchange anchors with others. Overall, as long as the majority of neighbors of each node remain unchanged within a time slot, PAFV copes well, and outperforms both GAS and RAS by a large margin. Only when the network is too dynamic for the decision of a time slot to be effective, RAS begins to outperform others.

In PAFV we consider the setting of slot duration and the playback delay as given parameters. Nevertheless, there are multiple choices of the slot durations with corresponding number of slots for a fixed playback delay, leading to different overall distortions and beacon overheads. We hence evaluate the effect of different slot durations and the number of slots to our simulation in Fig. 11. The playback delay is fixed, with different choices of the number of slots (and corresponding slot durations). The overall

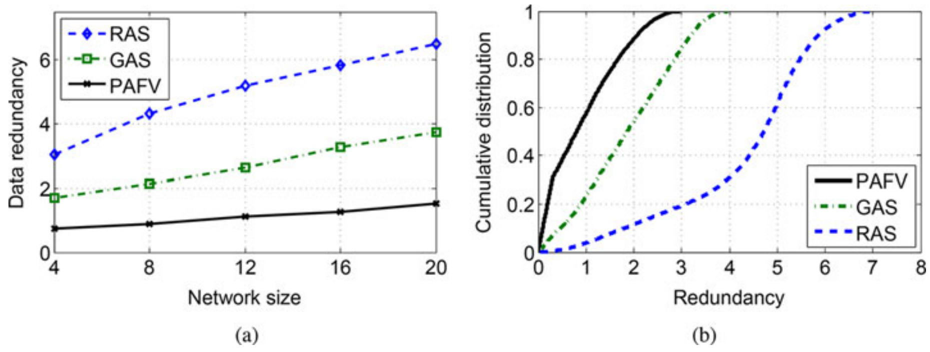


Fig. 12. Data redundancy. (a) Data redundancies versus network size. (b) Cumulative distribution of redundancy.

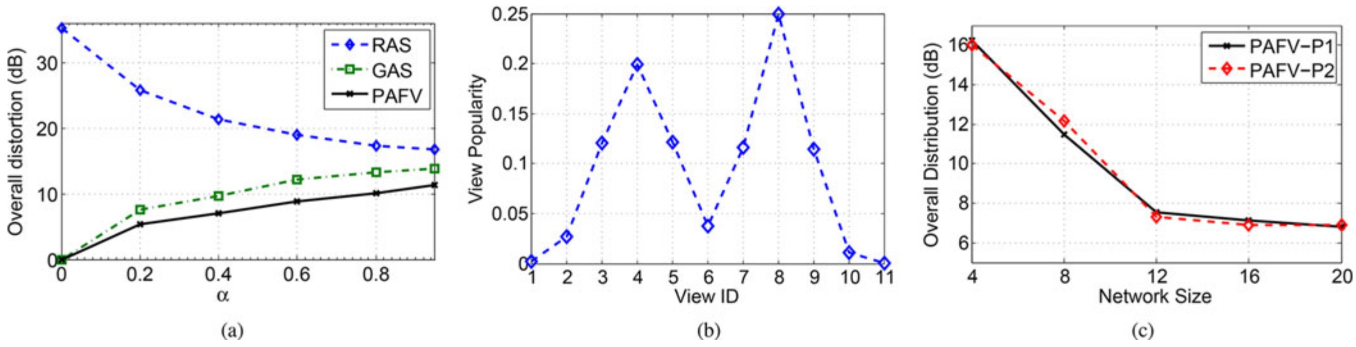


Fig. 13. Impact of view popularities. (a) Overall distortion versus $\alpha$ in (21). (b) View popularities generated using (24). (c) Overall distortion versus different popularity distributions.

TABLE V
VIEW PROPERTIES IN EXPERIMENT (B = BALLOONS, K = KENDO)

| View ID | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| Popularity | 0.2161 | 0.3602 | 0.2161 | 0.1297 | 0.0778 |
| Optimal PSNR (dB) (B) | 45.9807 | 33.6689 | 45.7871 | 34.1529 | 45.7530 |
| Optimal PSNR (dB) (K) | 46.7693 | 34.3330 | 46.7769 | 36.0556 | 46.5155 |

TABLE VI
EXPERIMENT PARAMETERS

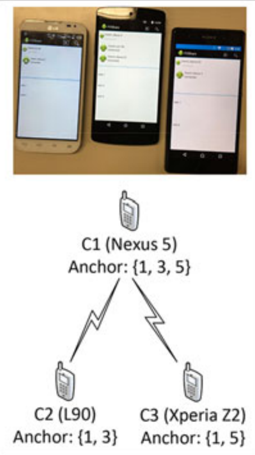
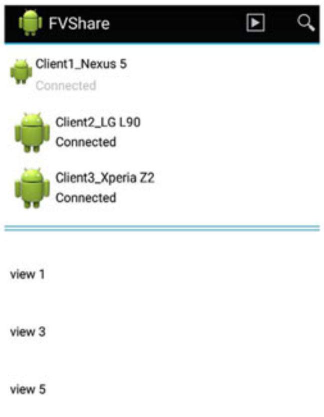
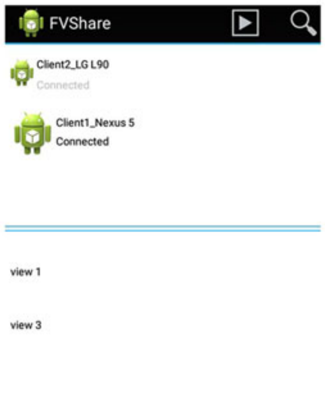
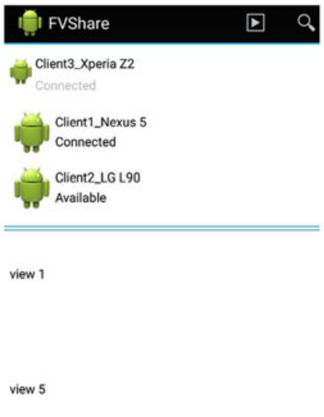
| Parameters | Value |
|---|---|
| clients | C1 (Nexus 5), C2 (LG L90), C3 (Sony Xperia Z2) |
| Connection standard | WiFi Direct |
| Connectivity | $\{1,2\}$,$\{1,3\}$ |
| Group owner (GO) | C1 |
| Transport layer | UDP Multicast |
| Pulling capacities | $r_1 = 3, r_2 = 1, r_3 = 0$ |
| Sharing capacities | $r'_1 = 1$ |
| Testing sequences | Balloons; Kendo |
| View set | $\mathcal{U} = \{1,2,3,4,5\}$ |
| Anchor set | $\mathcal{A} = \{1,3,5\}$ |
| Initial anchors | $\mathcal{A}_1 = \{1,3,5\}, \mathcal{A}_2 = \{1,3\}, \mathcal{A}_3 = \{1,5\}$ |
| Resolution | $512 \times 384$ |
| GoP format | H.264, 30 frame/s |
| GoP type | IPPP (QP: I 25, P 25) |
| Popularity setup | $j = 2, \alpha = \beta = 0.6$ in Eq. (21). |

distortion decreases as the number of slots increases, as more high-efficiency NC packets can be shared. Such decrease slows down after most of the popular anchors have been recovered at most nodes. In the meantime, the total number of beacons increases linearly, as each node sends a beacon in each slot. With such analysis, we set 3 slots in our simulation scenario. The optimal setting of slots in other scenarios should depend on the primary and secondary channel bandwidth, as well as client density, so that most of the popular anchors can be recovered through such setting.

The average number of redundancy experienced at a node versus network size is plotted in Fig. 12(a). The means of both primary and secondary channel bandwidths are increased to 20 units. As the network grows, PAFV maintains the least amount of duplicates among all three algorithms. The duplication experienced by RAS and GAS is higher, and grows proportionally with the network size. In GAS many nodes tend to pull popular anchors from the server, while in RAS anchors are randomly pulled by neighboring nodes. Fig. 12(b) shows the redundancy distribution. In PAFV duplicates may still occur when multiple nodes try to supply the same anchor to their common neighbors.

We evaluate the performance of PAFV with different popularity distributions in Fig. 13(a). We can see that when a small number of views are extremely popular, both PAFV and

TABLE VII
EXPERIMENT NETWORK SETUP AND THE SCREENSHOTS OF INITIAL AND RESULTANT ANCHOR CONDITIONS AT CLIENTS

| Devices & network | Client 1 (C1) | Client 2 (C2) | Client 3 (C3) |
|---|---|---|---|
|  C1 (Nexus 5)<br>Anchor: {1, 3, 5}<br>C2 (L90)<br>Anchor: {1, 3}<br>C3 (Xperia Z2)<br>Anchor: {1, 5} | FVShare<br>Client1_Nexus 5<br>Connected<br>Client2_LG L90<br>Connected<br>Client3_Xperia Z2<br>Connected<br>view 1<br>view 3<br>view 5 | FVShare<br>Client2_LG L90<br>Connected<br>Client1_Nexus 5<br>Connected<br>view 1<br>view 3 | FVShare<br>Client3_Xperia Z2<br>Connected<br>Client1_Nexus 5<br>Connected<br>Client2_LG L90<br>Available<br>view 1<br>view 5 |

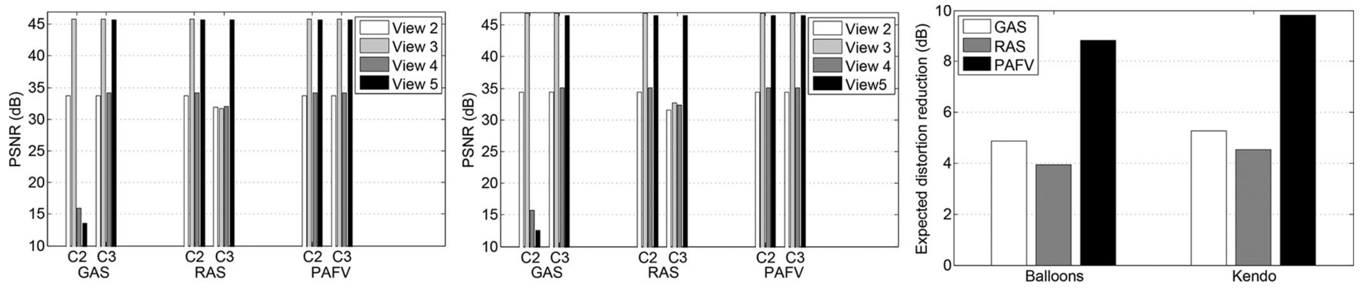


Fig. 14. Resultant PSNR of different schemes, and the overall expected distortion reduction. (a) Balloons. (b) Kendo. (c) Overall expected distortion reduction.

GAS perform well, while RAS suffers from pulling unpopular anchors. As the popularity distribution becomes uniform, the overall distortion of RAS decreases as more anchors become equally important. Meanwhile for PAFV and GAS, missing anchors become more important, leading to increased distortions. Popularity-based anchor prioritization keeps their performance better than that of RAS, while network coding ensures the lowest overall distortion for PAFV at all time. To evaluate PAFV's sensitivity to different popularity distributions, we generate another view popularity distribution for each view $i, 1 \le i \le |\mathcal{U}|$ by

$$p_i = \frac{1}{\sigma_1\sqrt{2\pi}} e^{-\frac{(x-\mu_1)^2}{2\sigma_1^2}} + \frac{1}{\sigma_2\sqrt{2\pi}} e^{-\frac{(x-\mu_2)^2}{2\sigma_2^2}}. \tag{23}$$

We then normalize each $p_i$ by

$$p_i' = \frac{p_i}{\sum_{i=1}^{|\mathcal{U}|} p_i} \tag{24}$$

and use $p_i'$ as the popularity of view $i$. Fig. 13(b) shows the generated view popularities with $\mu_1 = 4, \sigma_1 = 1, \mu_2 = 8, \sigma_2 = 0.8$. Denote the popularity generate by (21) as P1, and that generated by (24) as P2. We show the overall distortion for the two popularities against different network sizes in Fig. 13(c). It is clearly that PAFV is insensitive to different popularity distributions.

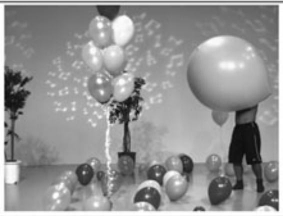
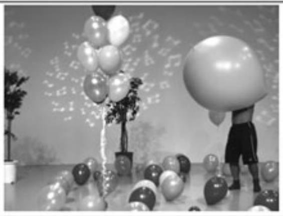
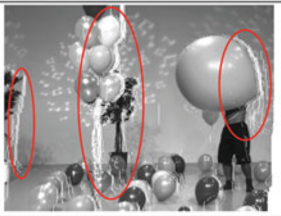
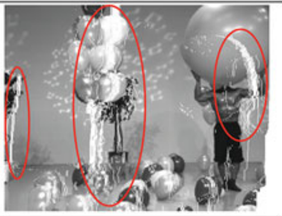
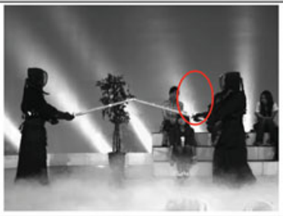
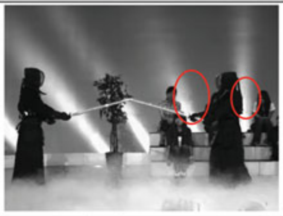
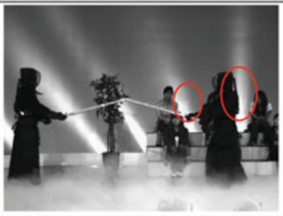
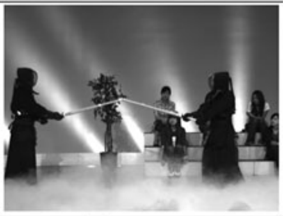
## VI. EXPERIMENTAL VALIDATION

Apart from simulation, we have also conducted an android-based experiment to validate the design of PAFV. The proof-of-concept experiments mainly serve to demonstrate the implementability and working of PAFV. We first describe network setup and video sequences in Section VI-A. We then present experiment results in Section VI-B.

### A. Experimental Setup

In each of the testing sequences Balloons and Kendo [35], there are 7 views. We choose view set $\{1, 3, 5\}$ to be the anchor set, and $\{2, 4\}$ the virtual view set. View set $\{0, 6\}$ is not used. Anchors views are generated by compressing raw video to $512 \times 384$ H.264 format. The optimal quality (highest PSNR) of an anchor is simply the PSNR after compression, while that of a virtual view is the PSNR after rendering from left and right nearest anchors. For view popularity, we use

TABLE VIII
SAMPLE PF VIEWS 2–5 OF TESTING SEQUENCES RENDERED AT C2 AND C3 AFTER DIFFERENT SCHEMES

| | View 2 | View 3 | View 4 | View 5 |
|---|---|---|---|---|
| C2 GAS Balloons |  |  |  |  |
| C3 RAS Kendo |  |  |  |  |
| C2, C3 PAFV Balloons | | | | |
| C2, C3 PAFV Kendo | | | | |

(20) and (21), and set $j = 2$ and $\alpha = \beta = 0.6$. View popularities and optimal PSNRs of the two sequences are summarized in Table V, with compression parameters and properties listed in Table VI.

There are three android phone clients in the network: Client 1 (C1, Nexus 5), Client 2 (C2, LG L90) and Client 3 (C3, Sony Xperia Z2). The three clients (which might be part of a larger network) form a *Wi-Fi Direct group*, where C1 is the *group owner (GO)*. UDP MulticastSocket is used for both neighbor information exchange and data transmission. C1 has a primary channel bandwidth of 3 anchors and a secondary channel bandwidth of 1 anchor. C1 therefore has the complete anchor set $\{1, 3, 5\}$. C2 has a primary channel bandwidth of 1 anchor, while C3 has zero primary channel bandwidth (i.e., no connection to server).[9] C2 initially has anchor set $\{1, 3\}$, one pulled from the server, and the other shared from its own neighbors outside the group. C3 has the initial anchor set $\{1, 5\}$, both of which shared from outside group. As far as the group is concerned, C2 and C3 will not utilize their secondary channel bandwidth since they are not able to benefit their neighbors (i.e., C1). Experiment parameters are listed in Table VI, with initial network setup shown in Table VII.

We run PAFV, GAS and RAS on C1 to see which anchor it shares to others, and evaluate our problem objective, which is the resultant overall expected distortion reduction.

## B. Validation Results

As the device screenshots in Table VII indicates, C1 running GAS (second row) always chooses anchor $\{3\}$ to share, as it brings higher expected distortion reduction to the group than anchor $\{5\}$. Recall that C2 has a pulling capacity of 1. So if C2 pulls anchor $\{3\}$ from the server and $\{1\}$ is shared from elsewhere, then in GAS, C2 may choose to pull anchor $\{5\}$ instead, and enjoy a full anchor recovery. If on the other hand anchor $\{1\}$ is pulled, then C2 will not change its pulling strategy and therefore has no benefit from C1's sharing. C1 running RAS (third row) may share any one of the three anchors with equal probability. PAFV (bottom row), combines view 3 and 5 together before sharing the coded packet, leading to a full anchor recovery at both C2 and C3. Note that in our experiment, we only focus on the cases where C2 pulls anchor $\{1\}$, and RAS shares anchor $\{5\}$, to differentiate the results of three schemes. To

[9]Note that due to the limited view number in available testing sequences, our experiment is of small scale with extra limit on network capacities, and serves the proof-of-concept purpose. Given large-scale video sequences, our experiment can be easily scaled up to large networks.

summarize, GAS recovers anchor $(\{3\})$ for C3; RAS recovers anchor $(\{5\})$ for C2; PAFV can recover both missing anchors simultaneously.

Having analyzed the sharing strategies for each scheme, we then show the resultant view quality (in PSNR) of the two sequences in Fig. 14(a) and (b). Note that we omitted view 1 in result analysis since all the clients have view 1 available. From Fig. 14, we observe optimal view quality at both C3 in GAS scheme and C2 in RAS scheme, while for C2 in GAS and C3 in RAS, the view PSNR remains low. In PAFV both clients enjoys complete anchor recovery and therefore experience optimal view quality. Table VIII presents selected samples of the visual qualities of views 2–5 at C2 and C3. We observe that GAS, unable to recover the missing anchor 5 for C2, left C2 with very poor and unimproved visual quality (with defects circled). Likewise, RAS is unable to help C3.[10] PAFV meanwhile is able to recover both missing anchors at C2 and C3, resulting in the optimal visual quality in two clients.

The overall expected distortion *reduction* is shown in Fig. 14(c). The distortion reduction of each view is weighted with view popularity, and then summed over all the views in both C2 and C3. We can see that PAFV clearly results in much higher distortion reduction. In this particular experiment, the distortion reduction of PAFV equals to the sum of distortion reduction in GAS and RAS.

## VII. Conclusion

In free viewpoint video (FVV), users may at any time switch to any view rendered from available anchors. We have, for the first time, studied a novel collaborative wireless network for live FVV streaming to a group of users. The users pull some anchors from server via a primary channel. They then generate network-coded (NC) packets with some of their anchors and share them with their direct neighbors using a secondary channel.

As video distortion of views is closely related to the available anchors, we seek to minimize video distortion by jointly optimizing the anchors to pull and the combination of anchors to generate NC packet for sharing, given the bandwidth limitations. Having presented a general middleware for such network, we formulated the joint optimization problem and showed that it is NP-hard. To address it, we have presented an efficient and fully distributed algorithm called PAFV (Peer-Assisted Freeview Video), where each node independently and collaboratively codes and shares NC packets based on neighbor information. Extensive simulation and android-based video experiments show that PAFV outperforms other schemes, achieving substantially lower video distortion (often by 20–50%) with significantly less data redundancy (by as much as 70%).

[10]The experiment also confirms that a virtual view will have a much higher quality if anchors of both sides are present, even if they are further away.

## Appendix
## NP-Hard Proof

We reduce *maximum coverage* [32] to our problem. The maximum coverage problem is stated as: given an integer $k$ and a collection of sets $\mathcal{S} = \{\mathcal{S}_1, \mathcal{S}_2, \cdots, \mathcal{S}_m\}$, find a subset $\mathcal{S}' \subseteq \mathcal{S}$, such that $|\mathcal{S}'| \leq k$, and $|\bigcup_{\mathcal{S}_i \in \mathcal{S}'} \mathcal{S}_i|$ is maximized.

We first simplify our problem as follows. Suppose there is a single client in the network. The primary channel bandwidth is $k$; pulling any anchor has a cost of 1. Anchor packet $a$ covers $S_a$ set of views. Views have a uniform popularity. We define a simple distortion function $D(u, \mathcal{A})$ for the views, which takes value 1, if the view is not covered by any anchor $a$ in pulled anchor set $\mathcal{A}$, and 0 otherwise, i.e.

$$D(u, \mathcal{A}) = \begin{cases} 1, & \text{if } u \notin S_a \quad \forall a \in \mathcal{A} \\ 0, & \text{otherwise.} \end{cases} \tag{25}$$

With only one client, the only decision parameter is $I^a, \forall a \in \hat{\mathcal{A}}$. Our Objective (1) can be simplified to

$$\min_{I^a} \sum_{u \in \mathcal{U}} D(u, \mathcal{A}) \tag{26}$$

subject to

$$\sum_{a \in \hat{\mathcal{A}}} I^a \leq k. \tag{27}$$

With distortion function defined by (25), Objective (26) can be interpreted as minimizing the number of uncovered views, or maximizing the number of covered views. We then reformulate $\sum_{u \in \mathcal{U}} D(u, \mathcal{A})$ in terms of view coverage as $|\mathcal{U} \backslash \bigcup_{a \in \mathcal{A}} \mathcal{S}_a|$. Objective (26) is then equivalent to maximizing the number of covered views, i.e.

$$\max_{I^a} \left| \bigcup_{a \in \mathcal{A}} \mathcal{S}_a \right| \tag{28}$$

subject to Constraint (27).

The reduction from maximum coverage to our simplified problem can then be done by using the set cardinality constraint $k$ as the pulling bandwidth constraint, and creating the collection $\mathcal{S} = \{\mathcal{S}_a, \mathcal{S}_b, \cdots, \mathcal{S}_{|\hat{\mathcal{A}}|}\}$ such that $\mathcal{S}_a$ is the set of views covered by anchor $a$. Any algorithm that solves this max coverage can solve our problem, by a lookup of the mappings between selected sets $\mathcal{S}_a$ and their corresponding anchor $a$, and set $I^a = 1$ for each picked set $\mathcal{S}_a$. Therefore, the simplified problem is at least as hard as the max coverage problem. So our original problem, being more general with given popularity distribution and distortion function as well as NC generation involvement, is NP-hard.

## References

[1] D. Miao, W. Zhu, C. Luo, and C. W. Chen, "Resource allocation for cloud-based free viewpoint video rendering for mobile phones," in *Proc. ACM MM*, Nov. 2011, pp. 1237–1240.

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

[3] D. Tian, P.-L. Lai, P. Lopez, and C. Gomila, “View synthesis techniques for 3D video,” in *Proc. SPIE 7443, Appl. Digital Image Process. XXXIII*, Sep. 2009, vol. 7443, pp. 74430T–.

[4] G. Cheung, V. Velisavljevic, and A. Ortega, “On dependent bit allocation for multiview image coding with depth-image-based rendering,” *IEEE Trans. Image Process.*, vol. 20, no. 11, pp. 3179–3194, Nov. 2011.

[5] H. S. Lichte, H. Frey, and H. Karl, “Fading-resistant low-latency broadcasts in wireless multihop networks: The probabilistic cooperation diversity approach,” in *Proc. ACM MobiHoc*, Sep. 2010, pp. 101–110.

[6] X. Jiao *et al.*, “Minimum latency broadcast scheduling in duty-cycled multihop wireless networks,” *IEEE Trans. Parallel Distrib. Syst.*, vol. 23, no. 1, pp. 110–117, Jan. 2012.

[7] X. Zhang and K. Shin, “Delay-optimal broadcast for multi-hop wireless networks using self-interference cancellation,” *IEEE Trans. Mobile Comput.*, vol. 12, no. 1, pp. 7–20, Jan. 2013.

[8] M.-F. Leung and S.-H. G. Chan, “Broadcast-based peer-to-peer collaborative video streaming among mobiles,” *IEEE Trans. Broadcast.*, vol. 53, no. 1, pp. 350–361, Mar. 2007.

[9] Y.-F. Wen and W. Liao, “Minimum power multicast algorithms for wireless networks with a lagrangian relaxation approach,” *Wireless Netw.*, vol. 17, no. 6, pp. 1401–1421, Aug. 2011.

[10] A. A. Khalek and Z. Dawy, “Energy-efficient cooperative video distribution with statistical QoS provisions over wireless networks,” *IEEE Trans. Mobile Comput.*, vol. 11, no. 7, pp. 1223–1236, Jul. 2012.

[11] I. Caragiannis, M. Flammini, and L. Moscardelli, “An exponential improvement on the MST heuristic for minimum energy broadcasting in ad hoc wireless networks,” *IEEE/ACM Trans. Netw.*, vol. 21, no. 4, pp. 1322–1331, Aug. 2013.

[12] B. Zhang, S.-H. G. Chan, G. Cheung, and E. Chang, “LocalTree: An efficient algorithm for mobile peer-to-peer live streaming,” in *Proc. IEEE Int. Conf. Commun.*, Jun. 2011, pp. 1–5.

[13] W. Ren *et al.*, “Broadcasting in multi-radio multi-channel wireless networks using simplicial complexes,” *Wireless Netw.*, vol. 19, no. 6, pp. 1121–1133, Aug. 2013.

[14] *Report of the Subjective Quality Evaluation for MVC Call for Evidence* ISO/IEC JTC1/SC29/WG11, Jan. 2005.

[15] G. Cheung, A. Ortega, and T. Sakamoto, “Coding structure optimization for interactive multiview streaming in virtual world observation,” in *Proc. IEEE 10th Workshop Multimedia Signal Process.*, Oct. 2008, pp. 450–455.

[16] Y. Chen *et al.*, “The emerging MVC standard for 3D video services,” *EURASIP J. Adv. Signal Process.*, vol. 2009, no. 1, pp. 786015–, Jan. 2009.

[17] Z.-P. Deng, Y.-L. Chan, K.-B. Jia, C.-H. Fu, and W.-C. Siu, “Iterative search strategy with selective bi-directional prediction for low complexity multiview video coding,” *J. Vis. Commun. Image Representation*, vol. 23, no. 3, pp. 522–534, Apr. 2012.

[18] D. Ren, S.-H. G. Chan, G. Cheung, and P. Frossard, “Coding structure and replication optimization for interactive multiview video streaming,” *IEEE Trans. Multimedia*, vol. 16, no. 7, pp. 1874–1887, Nov. 2014.

[19] M. Solh, G. AlRegib, and J. M. Bauza, “3VQM: A vision-based quality measure for DIBR-based 3D videos,” in *Proc. IEEE Int. Conf. Multimedia Expo*, Jul. 2011, pp. 1–6.

[20] B. Macchiavello, C. Dorea, E. Hung, G. Cheung, and W. T. Tan, “Loss-resilient coding of texture and depth for free-viewpoint video conferencing,” *IEEE Trans. Multimedia*, vol. 16, no. 3, pp. 711–725, Apr. 2014.

[21] Z. Liu, G. Cheung, and Y. Ji, “Optimizing distributed source coding for interactive multiview video streaming over lossy networks,” *IEEE Trans. Circuits Syst. Video Technol.*, vol. 23, no. 10, pp. 1781–1794, Oct. 2013.

[22] Y. Yang *et al.*, “Stereotime: A wireless 2D and 3D switchable video communication system,” in *Proc. ACM MM*, Oct. 2013, pp. 473–474.

[23] C. De Raffaele and C. Debono, “Applying prediction techniques to reduce uplink transmission and energy requirements in mobile free-viewpoint video applications,” in *Proc. IEEE 2nd Int. Conf. Adv. Multimedia*, Jun. 2010, pp. 55–60.

[24] L. Toni, N. Thomos, and P. Frossard, “Interactive free viewpoint video streaming using prioritized network coding,” in *Proc. IEEE Int.Workshop Multimedia Signal Process.*, Sept. 2013, pp. 446–451.

[25] S. Y. El Rouayheb, M. Chaudhry, and A. Sprintson, “On the minimum number of transmissions in single-hop wireless coding networks,” in *Proc. IEEE Inform. Theory Workshop*, Sep. 2007, pp. 120–125.

[26] N. Aboutorab, P. Sadeghi, and S. Sorour, “Enabling a tradeoff between completion time and decoding delay in instantly decodable network coded systems,” *IEEE Trans. Commun.*, vol. 62, no. 4, pp. 1296–1309, Apr. 2014.

[27] Z. Liu, G. Cheung, J. Chakareski, and Y. Ji, “Multiple description coding & recovery of free viewpoint video for wireless multi-path streaming,” *IEEE J. Sel. Topics Signal Process.*, vol. 9, no. 1, pp. 151–164, Feb. 2015.

[28] D. Ren, S.-H. G. Chan, G. Cheung, H. V. Zhao, and P. Frossard, “Anchor view allocation for collaborative free viewpoint video streaming,” *IEEE Trans. Multimedia*, vol. 17, no. 3, pp. 307–322, Mar. 2015.

[29] J. Chakareski, “Transmission policy selection for multi-view content delivery over bandwidth constrained channels,” *IEEE Trans. Image Process.*, vol. 23, no. 2, pp. 931–942, Feb. 2014.

[30] *Technical specification group services and system aspects; multimedia broadcast/multicast service (MBMS) user services; stage 1 (release 6)* 3GPP TS.26.246 ver. 6.3.0 3rd Generation Partnership Project, Mar. 2006.

[31] S. Sorour and S. Valaee, “Completion delay minimization for instantly decodable network codes,” *CoRR*, Jan. 2012 http://dblp.uni-trier.de/rec/bib/journals/corr/abs-1201-4768.

[32] R. M. Karp, “Reducibility among combinatorial problems,” in *Proc. IBM Res. Symp. Ser.*, Mar. 1972, pp. 85–103.

[33] U. Feige, “Approximating maximum clique by removing subgraphs,” *SIAM J. Discrete Math.*, vol. 18, no. 2, pp. 219–225, Feb. 2005.

[34] D. Johnson and D. Maltz, “Dynamic source routing in ad hoc wireless networks, ” *Mobile Computing* Kluwer Int. Ser. in Eng. and Comput. Sci., T. Imielinski, and H. Korth, Eds., New York, NY, USA : Springer, 1996, vol. 353, pp. 153–181.

[35] Tanimoto Lab., Dept. of Inform. Eng., Nagoya Univ., Nagoya, Japan, *“MPEG-FTV test sequence download page,”* (2008). [Online]. Available: http://www.tanimoto.nuee.nagoya-u.ac.jp/~fukushima/mpegftv/, Accessed on: Nov. 25, 2015.

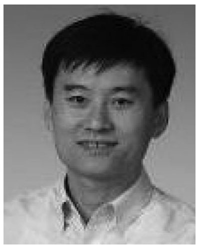
**Bo Zhang** (S’11) received the B.Sc. degree in computer science from Southampton University, Southampton, U.K., in 2006, the M.Sc. degree in data communication networks and distributed systems from the University College London, London, U.K., in 2007, and is currently working toward the Ph.D. degree in computer science and engineering at the Hong Kong University of Science and Technology, Hong Kong, China.

His research interests include wireless video broadcasting, multimedia networking, wireless network overlay, and peer-to-peer networks.

**Zhi Liu** (S’11–M’14) received the B.E. degree in computer science and technology from the University of Science and Technology of China, Hefei, China, in 2009, and the Ph.D. degree in informatics at the National Institute of Informatics and The Graduate University for Advanced Studies (SOKENDAI), Tokyo, Japan, in 2014.

He was a JSPS Research Fellow with the National Institute of Informatics and The Graduate University for Advanced Studies (SOKENDAI) from 2012 to 2014. From 2009 to 2014, he was a Research Assistant with the National Institute of Informatics and The Graduate University for Advanced Studies (SOKENDAI). He is currently an Assistant Professor with Waseda University, Tokyo, Japan. His research interests include wireless networks and video/image processing.

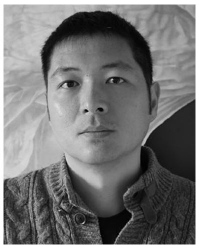
Prof. Liu is a Member of the IEICE. He was the recipient of the IEEE StreamComm 2011 Best Student Paper Award and VTC2014–Spring Young Researchers Encouragement Award.



**S.-H. Gary Chan** (S'89–M'98–SM'03) received the B.S.E. degree in electrical engineering with certificates in applied and computational mathematics, engineering physics, and engineering and management systems from Princeton University, Princeton, NJ, USA, in 1993, the M.S.E. degree in electrical engineering with a Minor in business administration from Stanford University, Stanford, CA, USA, in 1994, and the Ph.D. degree in electrical engineering from Stanford University in 1999.

He is currently a Professor in the Department of Computer Science and Engineering, Hong Kong University of Science and Technology (HKUST), Hong Kong, China. He is also the Chair of the Task Force on Entrepreneurship Education, HKUST. He has been a Visiting Professor and Researcher with Microsoft Research (2000–2011), Princeton University (2009), Stanford University (2008–2009), and the University of California at Davis, Davis, CA, USA(1998–1999). He was Undergraduate Programs Coordinator of Department of Computer Science and Engineering (2013–2015), Director of Sino Software Research Institute (2012–2015), Co-Director of the Risk Management and Business Intelligence Program (2011–2013), and Director of Computer Engineering Program (2006–2008) with HKUST. He has cofounded several startups with his team based on his research. His research interest includes multimedia networking, wireless networks, mobile computing and IT entrepreneurship.

Prof. Chan is a Member of Tau Beta Pi, Sigma Xi, and Phi Beta Kappa honor societies. He has been an Associate Editor of the IEEE Transactions on Multimedia (2006–2011) and a Vice-Chair of the Peer-to-Peer Networking and Communications Technical Sub-Committee of the IEEE ComSoc Emerging Technologies Committee (2006–2013). He is and has been Guest Editor of the ACM *Transactions on Multimedia Computing, Communications and Applications* (2016), the 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 the IEEE Consumer Communications and Networking Conference (2010), the Multimedia Symposium of IEEE GlobeCom (2007 and 2006), the IEEE International Conference on Communications (2007 and 2005), and the Workshop on Advances in Peer-to-Peer Multimedia Streaming at the ACM Multimedia Conference (2005). He was the recipient of industrial innovation awards in Hong Kong, Pan Pearl River Delta, and Asia-Pacific regions for various projects (2012–2015). He was the recipient of the Google Mobile 2014 Award (2010 and 2011) and the Silver Award of Boeing Research and Technology (2009). He was a William and Leila Fellow at Stanford University (1993–1994), and was the recipient of the Charles Ira Young Memorial Tablet and Medal and the POEM Newport Award of Excellence at Princeton (1993).



**Gene Cheung** (S'99–M'00-SM'07) received the B.S. degree in electrical engineering from Cornell University, Ithaca, NY, USA, in 1995, and the M.S. and Ph.D. degrees in electrical engineering and computer science from the University of California at Berkeley, Berkeley, CA, USA, in 1998 and 2000, respectively.

He was a Senior Researcher with Hewlett-Packard Laboratories Japan, Tokyo, Japan, from 2000 to 2009. He is now an Associate Professor with the National Institute of Informatics, Tokyo, Japan. He has been an Adjunct Associate Professor with the Hong Kong University of Science & Technology (HKUST), Hong Kong, China, since 2015. His research interests include image and video representation, immersive visual communication and graph signal processing.

Prof. Cheung has served as Associate Editor for the IEEE Transactions on Multimedia (2007–2011) and the DSP Applications column in the *IEEE Signal Processing Magazine* (2010–2014). He currently serves as Associate Editor for theIEEE Transactions on Image Processing(2015–present), the IEEE Transactions on Circuits and Systems for Video Technology (2016–present), the SPIE *Journal of Electronic Imaging* (2014–present), and the APSIPA*Journal on Signal & Information Processing* (2011–present), and as Area Editor for EURASIP *Signal Processing: Image Communication* (2011–present). He served as a Member of the Multimedia Signal Processing Technical Committee (MMSP-TC) in the IEEE Signal Processing Society (2012–2014), and a Member of the Image, Video, and Multidimensional Signal Processing Technical Committee (IVMSP-TC) (2015–2017). He is a coauthor of the recipient of the Best Student Paper Award in the IEEE Workshop on Streaming and Media Communications 2011 (in conjunction with the IEEE International Conference on Multimedia and Expo (ICME) 2011), the Best Paper Finalists in ICME 2011, the IEEE International Conference on Image Processing (ICIP) 2011, and ICME 2015, the Best Paper Runner-Up Award in ICME 2012, and the Best Student Paper Award in ICIP 2013.