# Motion-Prediction-Based Multicast for 360-Degree Video Transmissions

Yanan Bao, Tianxiao Zhang, Amit Pande, Huasen Wu, and Xin Liu
Department of Computer Science, University of California, Davis, CA 95616, USA
{ynbao, txzhang, pande, hswu, xinliu}@ucdavis.edu

*Abstract*—**360-degree video is on the cusp of going mainstream. Such videos have a large size (4-5× the size of a regular one). Since each viewer has to wear Head-Mounted Display (HMD), screen sharing is impossible when a group is watching the same content. Multiple parallel video-streams will be required to serve a group of viewers along the last mile (think of a family or a group of friends watching Super Bowl in their HMDs). This would end up quickly choking the entire network. In this paper, we present a scheme to optimize the network bandwidth using motion-prediction-based multicast to serve concurrent viewers. Based on empirical evaluation of more than 150 viewers watching our pool of sixteen 360-degree videos, we observe that most viewers follow similar motion patterns when watching the same video. We present a data-driven scheme for temporal prediction of viewer motion from previous states, and hence optimize the multicast bandwidth consumption by sending only the portion likely to be watched by a group of viewers. Our evaluations with real viewer motion traces show a bandwidth saving of over 50%, compared to full frame video multicast, and significant bandwidth reduction compared to unicast.**

## I. Introduction

360-degree video is on the cusp of going mainstream. Compared to traditional videos, streaming high quality 360-degree videos requires much more network bandwidth [1]. First, pixels from all directions are transmitted to viewers. If to maintain the same resolution of the viewing area as traditional videos, the size of a 360-degree video will be 4-5× larger [2], [3]. Second, the Head-Mounted Display (HMD) is worn close to the eyes. A higher-than-normal resolution is required for good user experience. Currently, 4K resolution is the minimal requirement for supporting 360-degree videos, which requires an average bitrate of 40-60Mbps [4]. In the future, 6K or even higher resolution may be required. Third, HMD screens cannot be shared among viewers even when they are watching the same video content together. Consider a group of friends watching the Super Bowl together – each person needs to wear an individual HMD. Thus the required bandwidth is multiplied by the number of viewers. Combined, these factors mean that 360-degree videos impose excessive demands on network bandwidth.

Motivated by this challenge, in this paper, we develop multicast transmission schemes for 360 degree videos in order to reduce bandwidth consumption. Specifically, we consider the scenario where multiple viewers in the vicinity watch the same 360-degree video simultaneously.

We are inspired by two unique features of 360-degree videos. First, like people watching TV or computer screens together nowadays, in many application scenarios of 360-degree videos, multiple viewers in the vicinity watch content simultaneously. Examples include family/friends watching sports and entertainment together, multiple team teleconferencing, virtual reality classrooms, etc.. Unlike traditional videos viewing, where screens are often shared, HMDs cannot be shared and each HMD requires the same content to be delivered separately. In this case, multicast is especially powerful and has the tremendous potential to reduce bandwidth consumption.

Second, current streaming schemes always transmit panorama pictures to viewers in a unicast manner. However, at any given time, a viewer watches only a portion of the 360-degree video, thus wasting the extra bits transmitted. If one could predict which portion of the video the viewer needs, one could transmit only a portion of the video, in order to reduce bandwidth consumption [5]. [5] illustrates that this approach is feasible for a single viewer; is it still possible for multiple viewers when they naturally have different motions?

To answer this question, we evaluate the viewer motion traces from a set of 153 viewers watching 16 video clips. We find that most viewers follow similar patterns when watching the same 360-degree video. In other words, viewers focus more on a certain common area within the whole viewing sphere most of the time. Therefore, in most cases, it may not be necessary to transmit entire frames to viewers. Instead, partial frames can be transmitted to further save bandwidth.

Based on these findings, we propose a novel framework that utilizes both multicast and unicast to transmit partial frames to a group of viewers in a bandwidth-efficient manner. In this framework, we first use motion prediction to estimate viewers' future viewing directions. Then based on the prediction result, we calculate the required portion of the viewing sphere for each viewer. Next, based on the channel conditions of viewers, we decide how to transmit the overlapped viewing area efficiently, using either one multicast or multiple unicasts. Last, we develop a model that captures how partial frame transmission affects video compression and its corresponding bandwidth overhead. In summary, this work has the following contributions:

- Our data analysis shows that for most of the video clips, users' viewing directions are closely correlated. Therefore, partial frame multicast has the potential to significantly reduce bandwidth consumption.

- We design a framework to transmit 360-degree videos to viewers using both multicast and unicast, based on viewer motion prediction. The framework also considers the bandwidth overhead of video compression caused by partial transmission.
- Using real traces of viewing motion, we evaluate the performance of the proposed framework. The results show over 2× bandwidth savings over traditional multicast.

The paper is organized as follows: We introduce the background of 360-degree video and present the problems in Sec. II. Sec. III describes the experiments conducted, as well as the findings from the collected data. Next, Sec. IV provides the design of the motion-prediction-based multicast system. The performance evaluation part (Sec. V) uses the collected real traces of viewer motion, and tests the bandwidth consumption and user experience of the proposed system. The discussion and lastly the related work are given before we conclude our work.

## II. Problem Statement

### A. Introduction to 360-degree Videos

In this work, we consider 360-degree video playback using HMDs such as Facebook Oculus and HTC Hive. A 360-degree video is made by recording a real-world panorama or by animation. For the captured real-world 360-degree videos, the view in every direction is recorded simultaneously using an omnidirectional camera or a collection of cameras. Then, the images of different directions are stitched together using software. Each stitched image is called a frame. Such frames are normally transmitted to viewers as a regular video transmission [6]. Bandwidth consumption of 360-degree videos is typically 4-5×, compared with standard 2D videos.

With 360-degree videos, the pixels of a frame are distributed on a sphere surrounding the head of viewer. During playback (viewing), the HMD senses the viewing direction of the viewer. Based on the viewing direction (shown in Fig. 1), the corresponding portion of the sphere, defined by the Field Of View (FOV), is rendered and then displayed on the HMD. The FOV defines the extent of the observable world that can be seen at one time. For Oculus DK2, its horizontal, vertical, and diagonal FOVs are $110°$, $90°$, and $120°$, respectively [7]. This area covers about 20% of the whole sphere.

For a given device, the distance between the eyes of the viewer and the display is fixed. Hence the horizontal, vertical, and diagonal FOVs are fixed, and the viewpoint and roll angle (the $Z$ axis rotation in Fig. 1) decide the area of the sphere to be seen by the viewer. Note that the Euler's angles (pitch, yaw, roll) are corresponding to the axes $(X, Y, Z)$ that the head rotates around, shown in Fig. 1. For simplicity, we subsequently refer to them as $X$, $Y$ and $Z$ angles.

The FOV depends on the position of the viewer's head and is only a small portion of the sphere. If the system knows the position before transmission, then it only needs to transmit part of the frame, which reduces the required bandwidth. In [5], the authors show that viewer motion can be well predicted in a time interval of 0.1s-0.5s. This motivates the use of motion-prediction-based partial frame transmission to effectively save bandwidth consumption.

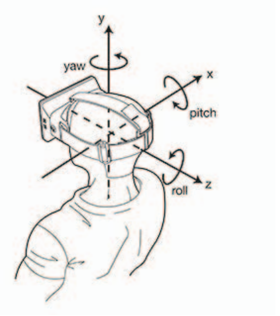


Fig. 1. The three angles [8]

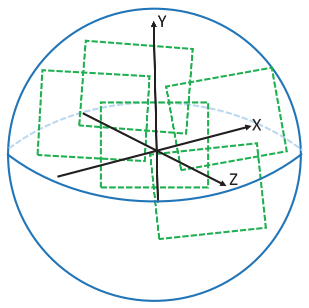


Fig. 2. Multiple views on the sphere

### B. Suitability of Multicast

Multicast has unique potential for 360-degree video transmissions: compared with traditional videos, 360-degree videos could have more screens watching the same content at the same time, as illustrated in Fig. 2. This is because there is no longer a common screen that can be shared among family members, conference attendees, or students in a classroom; each individual needs to wear an individual HMD to watch the same content. For this reason, the number of concurrent transmissions for the same content will be much larger, even in a small geographical area. This makes multicast a highly suitable technique to address the high network demand.

### C. Challenges

In this work, our objective is to design a bandwidth-efficient transmission scheme for 360-degree videos that combines multicast and unicast. In order to achieve bandwidth-efficient partial frame multicast, we need to address four main problems: **1)** How to predict viewer motion in the near future? Because there are sensor delay at the client, transmission delay between the server and the client, and frame processing delay at the server/client, the viewer motion has to be predicted in a range of 0.1s-0.5s, depending on different networks and implementations. This range is several times longer than the local delay at the client, considered by [9], [10]. Therefore, the prediction is more difficult and error-prone. **2)** How to guarantee user experience given that motion prediction is not perfect? In order to handle the prediction errors, we have to transmit more than the predicted FOV. In this work, we consider adding margins to the predicted FOV. **3)** How to transmit the data to viewers in a group in a bandwidth efficient manner? When certain viewing area on the sphere is required by multiple viewers, one needs to judiciously decide whether to use one multicast or multiple unicasts. **4)** How does partial transmission affect video compression, and how much is the corresponding overhead? We address these four questions in our designed framework (Sec. IV).

In [5], the authors consider transmitting partial 360-degree video frames to a **single viewer**. In comparison, we study how to efficiently transmit a 360-degree video to multiple concurrent viewers via **multicast**. We demonstrate the feasibility of multicast partial frames and design a partial frame multicast scheme. Furthermore, we develop an enhanced motion prediction model that reduces large prediction errors; we design a more efficient way to introduce margins to guarantee user experience; we develop a model that considers the impact of video compression.

## III. Data Description and Analysis

### A. Data Description

We use the data collected on [5]. We summarize the collected data here and refer readers to [5] for additional details. When a subject is watching a video, his or her motion is recorded and logged. The motion includes 3 degrees of freedom, pitch, yaw, and roll (i.e., $X$, $Y$, and $Z$ angles). When wearing a HMD, the viewer's initial position defines the zero degree for pitch, yaw and roll. Each dimension is denoted by an angle ($-180°$ to $180°$).

**360-degree Video Contents and Motion Measurements:** We download 16 clips of 360-degree video from YouTube and cut each of them into 30 seconds. Among the 16 videos, 14 are of 4K resolution, one is of 2K resolution, and one 1080P. Most videos have 30 frames per second. The 16 selected videos cover a variety of popular 360-degree video content available from the internet: 7 videos have sports content, e.g., basketball, boxing, soccer, skiing, etc., 4 videos contain landscape, e.g., Grand Canyon and tropical rain forest, and the other 5 videos contain entertainment activities, such as thrill rides and kite flying. In terms of camera movement, half of the videos have the camera fixed when shooting the video, while cameras of the other half videos move throughout the playback time. The video clips and sample motion data can be found at http://360videoexp.com/.

**Subjects:** In total, 153 volunteers joined the experiment, where 35 of them watched all 16 video clips and 118 of them watched 3∼5 randomly-selected video clips. According to ITU-R BT.500-11 subjective assessment standard [11], 15 subjects would be enough for subjective quality evaluation. The age distribution is as follows: 10∼20 (36%), 20∼30 (54%), 30∼40 (4%), 40∼50 (4%), 50∼60 (2%). 38% of the volunteers are female, and 34% of the volunteers wear glasses.

**Data Preprocessing:** The overall collected data include 985 views from the subjects, which generates 8.2 hours of recorded viewer motion. Our software collects 7∼9 samples per second, with variable sampling intervals. Therefore, we use linear interpolation to generate 60 samples per second from the raw data. This leads to about 1773000 total samples.

### B. Data Analysis

From the collected data, we find that most viewers have similar motion patterns when watching the same video. For example, in Fig. 3, we plot the heatmap of viewers' motions in $Y$-angles, i.e. the horizontal rotation, for the 16 videos. Each

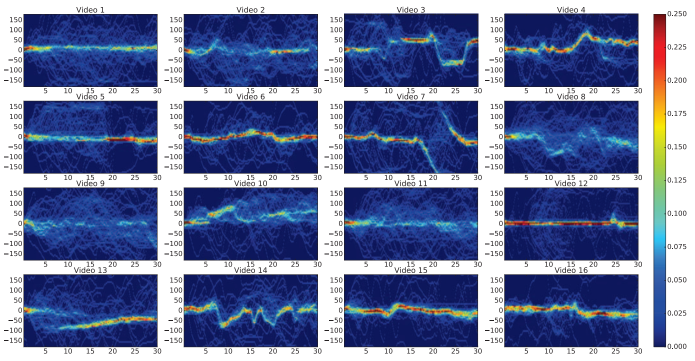


Fig. 3. Heatmap for $Y$-angles

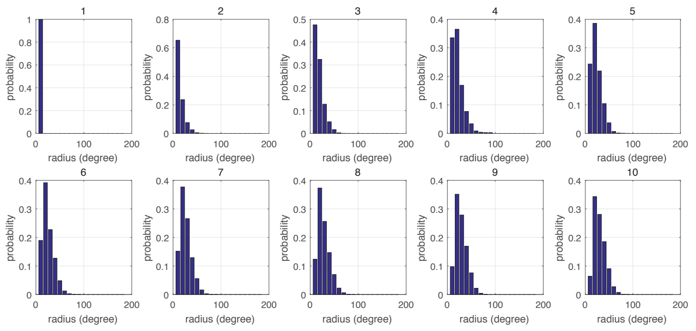


Fig. 4. The radius distribution in a multicast group (group size 1 to 10)

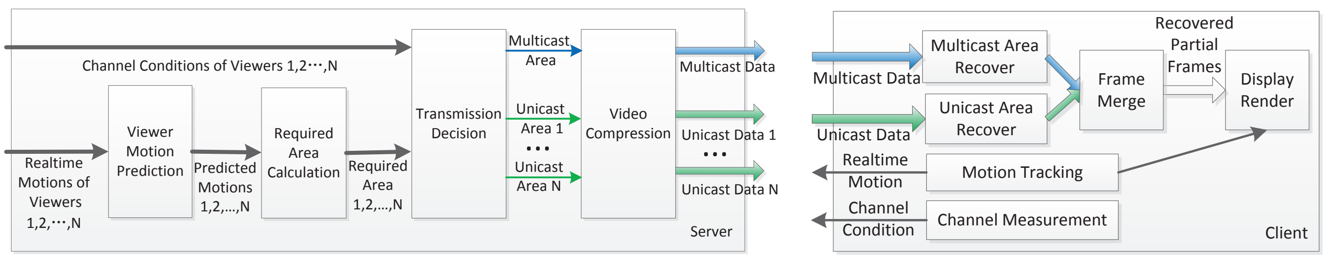
video is viewed by an average of 62 users, with a minimum of 47, and a maximum of 85. The X-axis corresponds to the video playout time and Y-axis corresponds to the degree of angles at any time. We can see that the $Y$-angles have dense concentrations for each of the 16 videos. This concentration illustrates the most common viewing angles of the users. For example, we can see for the 7th video, most of the viewers rotate 360° (given 0° to be the viewing angle at the beginning of playout time). Note that there are also videos whose viewer motion is less focused, for example, the 8th, 9th, 11th videos.

Furthermore, in Fig. 4, we draw the histogram of the radius of the smallest circle that covers the groups of viewpoint on the sphere, which reveals the divergence level of users' viewpoints. This figure shows that users' viewpoints are concentrated in a small area most of the time. For example, given 10 viewers, there is still 69% of time that the radius is less than 30°.

The above data analysis confirms our conjecture that whole frame transmission is often not necessary, and partial frames can be multicasted to reduce bandwidth consumption.

## IV. Motion-Prediction-Based Multicast

In this section, we design a motion-prediction-based multicast system. By tracking past viewer motions and channel conditions, the system predicts the viewer motion in next epochs, and decides the transmission scheme for each block of pixels. One block typically has $8 \times 8$ pixels or $16 \times 16$ pixels.



(a) Server side (b) Client Side

Fig. 5. The framework of 360-degree video multicast system

## A. System Framework

Fig. 5(a) and Fig. 5(b) show the design of the server and the client, respectively. Consider $N$ viewers in the system watching the same 360-degree video. The server mainly includes four modules: 1) viewpoint prediction based on each viewer's historical data; 2) required area calculation for each viewer given a pixel loss ratio constraint; 3) multicast or unicast decision for each single block, based on channel conditions; 4) video compression for the blocks before transmission.

On the client side, real-time viewer motion and network channel condition are tracked and sent to the server. The client receives multicast and unicast data, recovers multicast/unicast area with video decompression, and then merges them together. For unicast data, the client can request retransmission if packet loss occurs. For multicast data, there is no feedback or retransmission, and the server multicasts data with forward error correction (FEC) to make sure the data are received correctly by all the viewers. The overhead of FEC will also be considered in the bandwidth consumption.

## B. Motion Prediction

Based on a viewer's current and previous motions, the system needs to predict his or her future position. Each position includes three angles: $X$, $Y$, $Z$-angles. As shown in Fig. 6, for each of the three angles, the prediction model takes $[\theta_{t-n\Delta}, ..., \theta_{t-2\Delta}, \theta_{t-\Delta}, \theta_t]$ as inputs, and outputs $\theta_{t+t_p}$ as the forecast. Here $n$ is the size of input samples, $\Delta$ is the sampling interval, and $t_p$ is the prediction window.

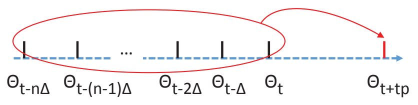


Fig. 6. The prediction timeline

We use neural networks as the regression models to predict the angles. Assume for an angle, the real value is $\theta_r$, and the predicted value is $\theta_p$. Then, the prediction error can be calculated with the following equation

$$Err(\theta_p, \theta_r) = |\text{mod}(\theta_p - \theta_r + 180^\circ,\ 360^\circ) - 180^\circ|. \quad (1)$$

In order to handle the periodical problem of angles, we use the same technique mentioned in [5] to project angles to a location on a ring with the unit radius.

TABLE I
THE STATISTICS OF $Y$-ANGLE PREDICTION ERROR

| $T_r$ | 0.1s | 0.2s | 0.3s | 0.4s | 0.5s |
|---|---|---|---|---|---|
| NN (Mean) | 0.92 | 2.44 | 4.33 | 6.33 | 8.40 |
| NN (RMSE) | 1.92 | 4.52 | 7.77 | 11.09 | 14.48 |
| NN (99th) | 6.54 | 17.25 | **30.54** | **44.03** | **57.59** |
| NN (99.9th) | 14.34 | 35.16 | **61.02** | **84.46** | **107.14** |
| Enhanced NN (Mean) | 1.00 | 2.91 | 5.26 | 7.69 | 10.12 |
| Enhanced NN (RMSE) | 1.93 | 4.73 | 8.19 | 11.70 | 15.26 |
| Enhanced NN (99th) | 6.33 | 16.58 | **29.26** | **42.00** | **55.01** |
| Enhanced NN (99.9th) | 13.89 | 34.19 | **58.01** | **79.99** | **101.40** |

Normally, Mean Square Error (MSE) is used as the cost function by regression models in the training step. However, in our work, we care more about the large errors, because we transmit FOV with margins. When small errors happen, the margins (discussed in Sec. IV-C) could cover them and no pixel would be lost. For this reason, 99 percentile and 99.9 percentile of the prediction errors are more helpful metrics in measuring the prediction accuracy. Compared with [5], we use an over-sampling method to further improve these metrics by 5%.

Specifically, we first train a neural network based on the initial training data. Second, we collect the training errors, and construct an updated training dataset by over-sampling the cases with large training errors. Finally, we train an enhanced neural network based on the updated training data.

As discussed in [5], we consider $t_p$ in the range from 0.1s to 0.5s, a reasonable prediction window that includes (multiple) transmission delays, video coding/decoding time, processing delay, etc.. We select 10 past samples (i.e., $n = 10$) as inputs with step size $\Delta = 0.1$s, approximately the recording interval of the motion tracking system. The hidden layer of the neural network has 5 neurons. We over-sample the data that gives the top 10% errors 10 times. 50% of the 1773000 samples are used to train a neural network, and the rest 50% are used for accuracy evaluation.

With these settings, a subset of the 99 percentiles and the 99.9 percentiles can be improved by more than 4%, as highlighted in Table I. In the table, **NN** is short for neural networks, **Enhanced NN** stands for our optimization with over-sampling, and the five columns show the statistics about the prediction errors for 0.1s to 0.5s in the future. Since over-sampling gives more weights to the large error case, the overall mean square error or mean error may increase.

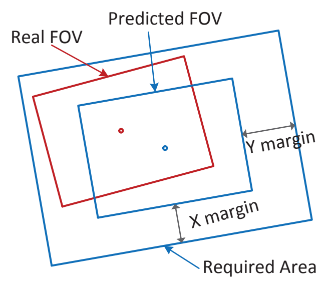


Fig. 7. An illustration of the required area

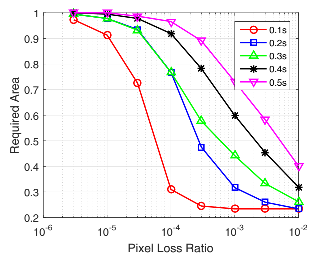


Fig. 8. The minimal required area for different PLR

TABLE II
MARGIN V.S PIXEL LOSS RATIO FOR A SINGLE VIEWER.

| Y margin \ X margin | $5^\circ$ | $10^\circ$ | $15^\circ$ | $20^\circ$ | $25^\circ$ | $30^\circ$ |
|---|---|---|---|---|---|---|
| $5^\circ$ | 5.78e-3 | 4.46e-3 | 4.23e-3 | 4.17e-3 | 4.15e-3 | 4.14e-3 |
| $10^\circ$ | 3.56e-3 | 2.04e-3 | 1.73e-3 | 1.65e-3 | 1.63e-3 | 1.62e-3 |
| $15^\circ$ | 2.93e-3 | 1.32e-3 | 9.38e-4 | 8.25e-4 | 7.87e-4 | 7.72e-4 |
| $20^\circ$ | 2.72e-3 | 1.07e-3 | 6.53e-4 | 5.05e-4 | 4.47e-4 | 4.24e-4 |
| $25^\circ$ | 2.65e-3 | 9.89e-4 | 5.54e-4 | 3.88e-4 | 3.10e-4 | 2.72e-4 |
| $30^\circ$ | 2.63e-3 | 9.64e-4 | 5.24e-4 | 3.52e-4 | 2.65e-4 | 2.15e-4 |
| $35^\circ$ | 2.62e-3 | 9.60e-4 | 5.19e-4 | 3.45e-4 | 2.56e-4 | 2.03e-4 |
| $40^\circ$ | 2.62e-3 | 9.58e-4 | 5.16e-4 | 3.42e-4 | 2.53e-4 | 1.98e-4 |

## C. Required Area

Based on the prediction of $X$, $Y$, $Z$-angles, we obtain a predicted FOV for each viewer. Since the prediction is not 100% accurate, more than the predicted FOV needs to be transmitted to guarantee enough pixels are received by the viewer. As shown in Fig. 7, the predicted FOV is different from the real FOV, and a larger rectangle with both $X$-margin and $Y$-margin is defined as the required area. This area is going to be transmitted to the viewer. The larger the margins, the higher the probability that the transmitted area covers the real FOV, at the cost of additional data transmission. Therefore, by tuning the margin size, we can balance the tradeoff between the user experience and the required bandwidth. In this work, we use Pixel Loss Ratio (PLR) as the user experience metric, which is defined as the number of lost pixels v.s. the total number of pixels in the actual FOV.

In Table II, we show the relation between margins and the pixel loss ratio for a single viewer. The prediction window $t_p = 0.2$s. The table shows that the neural network prediction is pretty accurate, for given a small pixel loss ratio, only small margins are needed. For example, with $15^\circ$ $X$-margin and $15^\circ$ $Y$-margin, the pixel loss ratio is less than 0.1%. With $30^\circ$ $X$-margin and $40^\circ$ $Y$-margin, the loss ratio is less than 0.02%. Given the pixel loss ratio less than 0.1%, a search with granularity of $1^\circ$ shows that when $X$-margin is $12^\circ$ and $Y$-margin is $18^\circ$, the required area has the minimal size, 0.296 of the full sphere. This combination of margins will be used in the performance evaluation part (Sec. V).

In Fig. 8, we show the minimal required area given different PLRs, normalized by the full sphere, by optimizing the combination of $X$-margin and $Y$-margin. The results include prediction window from 0.1s to 0.5s.

## D. Multicast and Unicast

Transmission bandwidth consumption is related to the wireless channel conditions. Let $C_i^u$ be the bandwidth consumption of transmitting one unit of data to viewer $i$ in unicast, and $C^m$ be the bandwidth consumption in multicast transmission of one unit of data to all the viewers. We assume that $C_i^u$ and $C^m$ are known for all transmissions based on channel condition information provided by the viewers. Furthermore, we assume that packet loss is handled using retransmissions in unicast, and by FECs in multicast, and thus overhead is taken account in $C_i^u$ and $C^m$, respectively. In the following, we consider two methods *multicast only* and *smart multicast* to transmit blocks.

As its name suggests, *multicast only* uses multicast to transmit the union of all the required blocks to all viewers, regardless of whether a block is required by one viewer or multiple viewers. Because each viewer receives data of not only its own required area, but also other viewers' required areas, it provides additional protection against pixel loss due to erroneous motion predictions.

However, it is well known that multicast may suffer in a wireless network when users have highly diverse channel conditions. Specifically, one viewer with very poor channel conditions may significantly increase the bandwidth consumption of all. To address this issue, we propose *smart multicast* that is more bandwidth efficient, by incorporating both multicast and unicasts. Specifically, as illustrated by Fig. 9, the blocks required by a single viewer are unicasted to the viewer. Furthermore, for a block required by multiple viewers, we need to decide whether to multicast it or unicast it as follows: Assuming the block is required by a subset of viewers, denoted by $\mathbf{S}$. When the multicast cost $C^m$ is larger than the cost of $|\mathbf{S}|$ individual unicasts, i.e.,

$$C^m > \sum_{i \in \mathbf{S}} C_i^u, \tag{2}$$

the block is transmitted to the viewers using $|\mathbf{S}|$ unicasts; otherwise, the block is transmitted to the viewers using one multicast. We note that this scheme is selected for its simplicity, and more sophisticated hybrid schemes can be adopted here, e.g., grouping users with similar channel conditions together. Compared with *multicast only*, *smart multicast* has higher computation cost, since each viewer needs to be treated individually during the video compression step, as discussed in Sec. IV-E.

## E. Video Compression

In our prediction-motion-based multicast, we need to multicast or unicast a partial frame. Certain modifications on the video compression scheme are necessary to achieve this partial transmission. We discuss the high-level idea about these modifications and present a model to capture the impact on bandwidth consumption.

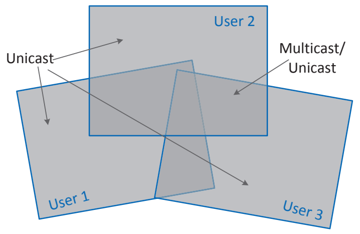


Fig. 9. Multicast/unicast in the overlapped area

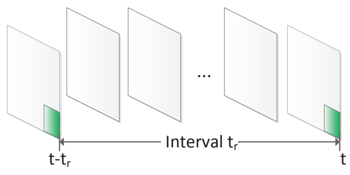


Fig. 10. Referencing frame interval for a certain block

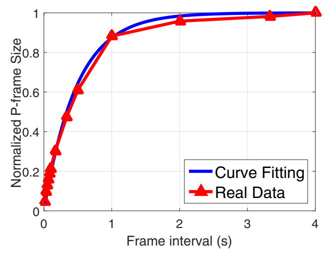


Fig. 11. Frame interval v.s. normalized P-frame size

Video compression is commonly used to reduce the bandwidth (bitrate) requirement. Within a group of pictures (GOP), there are usually I, P, B-frames. I-frames are independent of other frames, and only intra-frame compression is taken. P-frame references to a previous I or P-frame. B-frame is a bi-predictional frame, which references to a previous I or P-frame and a later P or I-frame. The referenced frames will reduce the new data that need to be compressed in the referencing frames. Since decoding B-frames needs to receive and decode the next P-frame first, baseline profile (only I and P-frames) is considered for maintaining low latency, as in [12], [13]. We adopt this model in this work.

Inherited from standard video compression, we consider block-based compression, where each block consists of $8\times8$ or $16\times16$ pixels. However, because frames are partially transmitted in our work, the motion compensation needs to be modified. If the current block references to a block that was not transmitted previously, the decoder cannot recover the current block. Therefore, the video compression for partial transmission cannot base on pervious full frames; for a block in the current frame, it can only reference to the transmitted blocks in the previous frames. In other words, the video compression coding needs to happen on the fly for each frame, because different viewers have different partial frames from earlier transmissions.

We could see that the computational overhead caused by such on-the-fly scheme satisfies our system requirement. According to [14], given a video segment with duration of one second, the average real-time transcoding time is less than 0.5 second based on the standard ISO/IEC 23009-1 of MPEG DASH. Since the segment can be split into smaller segments to be encoded by multiple processors, given a short GOP size, the delay of video compression is covered within the motion prediction window.

In this work, we construct a model to capture the cost of referencing to a block that is not in the previous frame. With this model, we are able to estimate the size of a compressed frame, whose blocks reference to different previous frames.

Let us consider a target block in the current frame $t$. If the most recent frame that has a block at the same position is $t - t_r$, as shown in Fig. 10, the search of the matched block can be taken in frame $t - t_r$. The size of the target frame compressed using a block at $t-t_r$ is related to the interval $t_r$. Usually, the longer the interval $t_r$, the larger the compressed target block, because of temporal correlation. Therefore, if we average over blocks in each frame, and frames over a video, the cost function should be a non-decreasing function of the interval $t_r$.

Based on the sixteen 360-degree videos, we did video compression experiments to obtain the relation between the interval $t_r$ and the cost. We choose different frame intervals, including [1,2,3,4,5,6,10,20,30,60,120,200,240]/60 seconds. First, given a video and a frame interval, we take the average size of all the P-frames generated by video compressions. Second, this average P-frame size is normalized by the result of 4 seconds, which we believe is long enough. We perform the normalization because videos have different resolutions and bitrates. Third, all points from the 16 videos are merged together, and the average values are shown in Fig. 11. Curve fitting shows the following result

$$cost(t_r) = 1 - 0.127^{t_r}. \tag{3}$$

As shown in Fig. 11, the curve fits the points very well. The result is also intuitive as the past information is decreasing exponentially. This cost model will be used in our simulation (Sec. V).

In the *smart multicast* scheme, due to motion compensation, we have several spheres of blocks to maintain. Specifically, on the server side, for each viewer, there is a unicast sphere to maintain. Apart from that, one common multicast sphere is maintained for all viewers. On the client side, each viewer maintains his or her own unicast sphere along with that common multicast sphere to reference. The transmission decision will affect which sphere to reference to, and therefore change the bitrate of the compressed video. However, taking this into consideration will make the transmission decision too complex. Therefore, it is left for future work.

## V. Performance Evaluation

Putting all the modules discussed together, in this section, we evaluate the performance of the proposed framework based on real viewer-motion traces. The performance metrics include both the bandwidth consumption and the pixel loss ratio. In our simulations, we include the results for the ideal case here, where the viewer motion in the short future is known a priori. This ideal case serves as the lower bound of the bandwidth consumption.

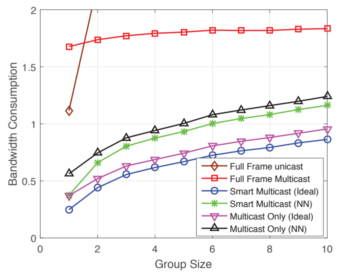


Fig. 12. Bandwidth consumption of homogeneous users

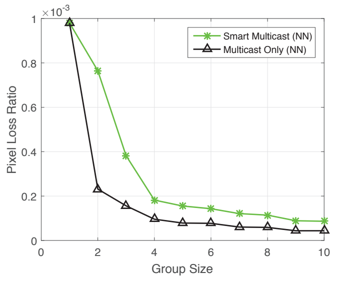


Fig. 13. Pixel loss ratio of homogeneous users

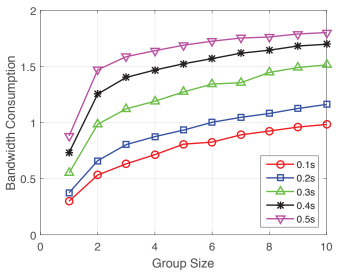


Fig. 14. Bandwidth consumption of different prediction windows

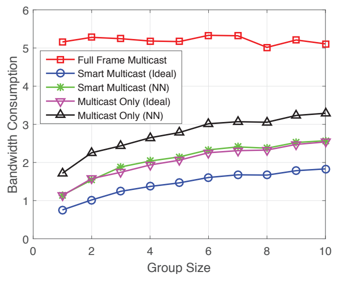


Fig. 15. Bandwidth consumption of heterogeneous users

## A. Simulation Settings

**System Setting:** The motion prediction window is set to be 0.2s, unless otherwise specified. As discussed in Sec. IV-B, the $X$-margin and $Y$-margin are set to $12°$ and $18°$, respectively, to achieve 0.1% PLR for a single viewer with the minimal required area.

**Channel Model:** We consider a small area with one access point (AP) and $N$ viewers. The AP can use both unicast and multicast to transmit data to the viewers. Assume the packet loss ratio of viewer $i$ is $p_i$, with retransmission, the cost of unicasting per packet to viewer $i$ is

$$C_i^u = \frac{1}{1-p_i}. \tag{4}$$

For multicast, because it needs to serve the worst viewer using FEC, we assume

$$C^m = r \max_{i \in N} C_i^u, \tag{5}$$

where $r \geq 1$ is a factor indicating the additional cost of FEC. We choose $r = 1.5$ in our simulation, which is a conservative estimate, according to [15]. Note that other channel models can be considered as well.

## B. Homogeneous Users

First, we consider a group of viewers with similar channel conditions. Assume for all the viewers, packet loss ratio $p_i$ is uniformly distributed in $[0, 0.2]$.

In Fig. 12, we plot the bandwidth consumption v.s. the group size. The bandwidth is normalized by the bandwidth consumption of full frame unicast with no packet loss. Note that without multicast, the bandwidth consumption of unicasting the video grows linearly with the number of viewers, which quickly goes out of the bound and thus is not included. Compared with full frame multicast, both *smart multicast* and *multicast only* can achieve more than 50% bandwidth saving. When the group size is larger than one, the gap between *smart multicast* and *multicast only* is relatively small. This is because, in *multicast only*, a larger area with much more pixels are multicasted to everyone in the group, therefore the video compression can be more efficient compared with unicast. From the figure, we can also see with ideal prediction, the bandwidth can be further reduced by 50%, which is the lower bound.

Fig. 13 plots the PLR v.s. the group size. When there is a single viewer in the system, both the transmission methods have 0.1% PLR. With more viewers, the average PLR is getting smaller. This is because the pixels at the multicasted area are received by all the viewers, and viewers benefit from each other. The pixel loss ratio of *multicast only* decreases fast than *smart multicast*, as more data are shared among viewers.

Given different prediction window, the consumed bandwidth of the *smart multicast* is plot in Fig. 14. With longer prediction window, the motion prediction is less accurate, and therefore larger margins are required to guarantee 0.1% PLR. This results in an increase of bandwidth consumption.

## C. Heterogeneous Users

In this part, we consider one viewer in the group has poor channel condition. Assume the viewer packet loss ratio is distributed uniformly in $[0.6, 0.8]$.

Fig. 15 shows the bandwidth consumption in this scenario. We can see the gap between the *smart multicast* and the *multicast only* is larger than that in the homogeneous case. This is because when the channel conditions are heterogeneous, using *smart multicast* could serve the viewers with good and poor channel conditions separately. Therefore, when viewers have similar channel conditions, *multicast only* is a good choice. Because it has similar bandwidth consumption, and lower PLR. Otherwise, *smart multicast* is more efficient, in the sense of bandwidth saving. The PLR performance with heterogeneous users is similar to Fig. 13, because transmitted areas do not depend on channel conditions.

## D. When and where does pixel loss occur?

In this part, we analyze when and where does pixel loss occur, particularly in the FOV, in different frames and among different viewers. In Fig. 16, we show the aggregated PLR in the FOV. From the figure, we can see all the PLRs higher than 0.02 are limited to a very small area at the four corners. It is possible the pixel loss may not even be noticed by a viewer. Therefore, even though sometimes pixels are lost, user experience may not be affected.

For all the transmitted frames, 96.63% frames have 0 pixel loss. For the other 3.37%, we plot their PLR distribution and normalize it by 3.37% in Fig. 17. It decays quickly and large PLR-s happen very rarely, which implies good user experience.

In Fig. 19, we show the distribution of PLR of different viewers, from group size 1 to group size 10. Most of viewers have similar PLR, which means the system is fair among

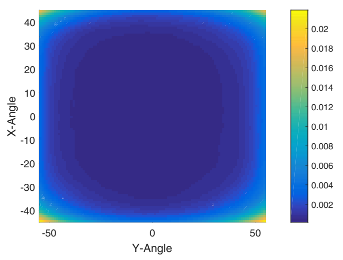


Fig. 16. Pixel loss ratio in the FOV

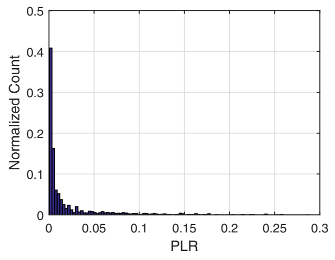


Fig. 17. Pixel loss ratio of different frames

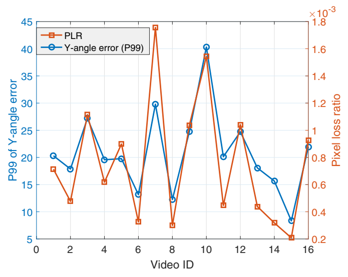


Fig. 18. Pixel loss ratio of different videos

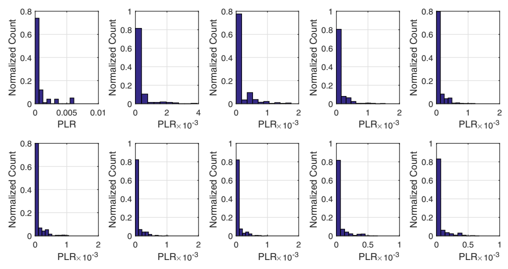


Fig. 19. Pixel loss ratio of different viewers (group size from 1 to 10).

viewers. Moreover, when there are more viewers in the system, the PLR distribution deviation decreases further.

For different videos, viewers behave differently. As shown in Fig. 18, different videos have different average PLRs. The difference could be as large as 2×. Last, we can see the PLR is highly related to 99 percentile of the $Y$-angle prediction error. It is our future work to study individualized models for different videos.

## VI. Discussions

As a starting point, the current motion traces are collected when the videos are played without audio. In practice, the audio provides another dimension to attract the viewers' attention. Particularly, when the videos are played with stereophonic audio, we expect that viewer motions would be more concentrated and the predictability will be increased. The impact of audio on prediction (specifically for stereophonic audio) is our future work.

In this paper, we mainly consider the last mile wireless channels as the bottleneck of 360-degree video transmission. However, for realtime video streaming, the wired network could also become a bottleneck for realtime video streaming, where no caching can be utilized. For example, in our lab, the wired speed test shows the end-to-end download speed is less than 100Mbps from a nearby major city. Given that a 4K 360-degree video has a bitrate of 40-60Mbps, at most two streamings can be supported simultaneously. The proposed framework and method also apply to the wired networks, (which is even simpler as the channel conditions are more stable in wired networks).

One issue in our proposed framework is the high computation for online video compression, especially if each unicast streaming is compressed individually. Compared to transmitting full video frames using simple multicast, where video compression is taken offline, online video compression requires much more computation. There are several ways to address this issue, for example using the storage to trade for computation, i.e., preprocessing different viewpoints' compressed frames offline.

For multicast, there is no retransmission. Therefore, some data could be lost even with FEC. The loss data will affect the later frames in a GOP. To alleviate this issue, we can use short GOP for multicast and long GOP for unicast.

## VII. Related Work

Multicast has been extensively studied for video distribution in networks [16], [17], [18], [19]. For example, in [19], unicast and overhearing are combined to disseminate downloaded packets within a group of smartphones. However, in these works, authors focus on traditional videos, while in our work we consider 360-degree videos, where partial frame transmission can significantly save bandwidth. Since the required data are dynamically changing due to the movement of viewpoints, 360-degree videos provide more challenges and opportunities.

In terms of transmitting 360-degree videos, the most related work is a protocol named *Dynamic Streaming* proposed by Facebook [20]. In this transmission protocol, the data around the FOV are transmitted with high resolution and the other areas are transmitted with low resolution. However, without viewer motion prediction, there is a high probability that viewers may watch the lower resolution part when they change their motion fast within one GOP. In our work, short term motion prediction reduces the pixel loss ratio to less than 0.1%, which gives user experience a guarantee. [6] proposes a region adaptive smoothing technique to save bits at the top and bottom areas in an equirectangular frame, which reduces up to 20% bitrate. However, the technique is limited to equirectangular projection of the 360-degree videos.

For viewpoint motion prediction, [5] illustrates the feasibility for a single user based on historical data. [9] considers motion prediction with constant angular velocity or acceleration to predict motion in the scale of 20-100 ms. However, the method is not data-driven, and the time scale it considers is not proper for video transmission. [10] presents a double exponential smoothing method to predict head position and rotation in the next 50 ms. Different from [10], we study viewpoint prediction in the range of 0.1s-0.5s, which is reasonable for video frame transmission from a server to a client. The longer prediction windows make prediction results more error-prone, which is addressed in our work by over-sampling and adding margins.

There is some work on optimizing the process of projecting 360-degree videos from 3D space to 2D space, for example equirectangular projection, cube-based mapping [21], and pyramid-based mapping [20]. Different methods have different 3D reconstruction quality and computing complexity. Compared with equirectangular projection, cube-based mapping and pyramid-based mapping can reduce a certain amount of the bandwidth consumption given the same viewing experience. In [22], a non-uniform spherical ray sampling method is proposed to give more priority to the important regions. These projecting methods could be utilized and adjusted in the implementation of our framework.

Apart from 360-degree videos, there are also free-viewpoint videos (FVV), where viewers are able to interact with the scene by navigating to different viewpoints. [23] studies a multi-view-plus-depth (MVD) representation to improve viewer's watching experience and increase visual quality. [24] considers multiview video streaming systems and improves efficiency in coding and content replication strategies. However, 360-degree videos are different from FVV in many aspects, such as video recording, frame projection, display rendering, video compression, which makes these techniques less applicable to 360-degree videos.

## VIII. Conclusion

In this work, we propose a multicast framework to transmit 360-degree videos to a group of viewers simultaneously. Based on the collected viewer motion traces, we find that in most cases, the viewers watching the same video focus their attentions on a certain common area. This similarity motivates our design of partial frame multicast, based on viewer motion prediction. Depending on viewer motion and channel conditions, adaptive multicast complemented by unicast is applied to transmit different blocks, such that the required bandwidth is significantly reduced without much user experience deterioration. Simulations based on real viewer motion traces show that our design saves more than 50% bandwidth compared with full frame multicast.

**Acknowledgments:** The work was partially supported by NSF through grants CNS-1547461; CNS-1457060; CCF-1423542; CRII-1464421.

## References

[1] M. Weldon, "The rational exuberance of 5G," available at https://www.bell-labs.com/var/articles/rational-exuberance-5G/.

[2] S. Hollister, "Youtube's ready to blow your mind with 360-degree videos," available at http://gizmodo.com/youtubes-ready-to-blow-your-mind-with-360-degree-videos-1690989402.

[3] K. Leswing, "Virtual reality is coming to youtube; here's how to watch it," available at http://www.ibtimes.com/virtual-reality-coming-youtube-heres-how-watch-it-1949250.

[4] N. Kraakman, "The best encoding settings for your 4K 360 3D VR videos + free encoding tool," available at http://www.purplepillvr.com/best-encoding-settings-resolution-for-4k-360-3d-vr-videos/.

[5] Y. Bao, H. Wu, T. Zhang, A. Ramli, and X. Liu, "Shooting a moving target: Motion-prediction-based transmission for 360-degree videos," in *Big Data (Big Data), 2016 IEEE International Conference on*. IEEE, 2016.

[6] M. Budagavi, J. Furton, G. Jin, A. Saxena, J. Wilkinson, and A. Dickerson, "360 degrees video coding using region adaptive smoothing," in *Image Processing (ICIP), 2015 IEEE International Conference on*. IEEE, 2015, pp. 750–754.

[7] P. R. Desai, P. N. Desai, K. D. Ajmera, and K. Mehta, "A review paper on oculus rift-a virtual reality headset," *International Journal of Engineering Trends and Technology (IJETT)*, vol. 13, no. 4, 2014.

[8] V. Oculus, "Oculus rift," *Available at http://www.oculusvr.com/rift*, 2015.

[9] S. M. LaValle, A. Yershova, M. Katsev, and M. Antonov, "Head tracking for the oculus rift," in *Robotics and Automation (ICRA), 2014 IEEE International Conference on*. IEEE, 2014, pp. 187–194.

[10] J. J. LaViola, "Double exponential smoothing: an alternative to kalman filter-based predictive tracking," in *Proceedings of the workshop on Virtual environments 2003*. ACM, 2003, pp. 199–206.

[11] I. Recommendation, "500-11,methodology for the subjective assessment of the quality of television pictures, recommendation ITU-R BT. 500-11," *ITU Telecom. Standardization Sector of ITU*, 2002.

[12] J. Wang, "Chitchat: Making video chat robust to packet loss," Ph.D. dissertation, Massachusetts Institute of Technology, 2010.

[13] I. E. Richardson, *H. 264 and MPEG-4 video compression: video coding for next-generation multimedia*. John Wiley & Sons, 2004.

[14] E. Baik, A. Pande, Z. Zheng, and P. Mohapatra, "Vsync: Cloud based video streaming service for mobile devices," in *IEEE INFOCOM 2016*, April 2016, pp. 1–9.

[15] A. RF, "Link budget analysis: Error control & detection," available at http://atlantarf.com/Error_Control.php.

[16] T. Turletti and J.-C. Bolot, "Issues with multicast video distribution in heterogeneous packet networks," in *In Proceedings of the Sixth International Workshop on Packet Video*. Citeseer, 1994.

[17] H. Ma and K. G. Shin, "Multicast video-on-demand services," *ACM SIGCOMM Computer Communication Review*, vol. 32, no. 1, pp. 31–43, 2002.

[18] W. Wei and A. Zakhor, "Multipath unicast and multicast video communication over wireless ad hoc networks," in *Broadband Networks, 2004. BroadNets 2004. Proceedings. First International Conference on*. IEEE, 2004, pp. 496–505.

[19] L. Keller, A. Le, B. Cici, H. Seferoglu, C. Fragouli, and A. Markopoulou, "Microcast: cooperative video streaming on smartphones," in *Proceedings of the 10th international conference on Mobile systems, applications, and services*. ACM, 2012, pp. 57–70.

[20] D. P. Evgeny Kuzyakov, "Next-generation video encoding techniques for 360 video and VR," available at https://code.facebook.com/posts/1126354007399553/next-generation-video-encoding-techniques-for-360-video-and-vr/.

[21] ——, "Under the hood: Building 360 video," available at https://code.facebook.com/posts/1638767863078802/under-the-hood-building-360-video/.

[22] J. Lee, B. Kim, K. Kim, Y. Kim, and J. Noh, "Rich360: optimized spherical representation from structured panoramic camera arrays," *ACM Transactions on Graphics (TOG)*, vol. 35, no. 4, p. 63, 2016.

[23] A. Hamza and M. Hefeeda, "Adaptive streaming of interactive free viewpoint videos to heterogeneous clients," in *Proceedings of the 7th International Conference on Multimedia Systems*. ACM, 2016, p. 10.

[24] H. Huang, B. Zhang, S.-H. G. Chan, G. Cheung, and P. Frossard, "Coding and replication co-design for interactive multiview video streaming," in *INFOCOM, 2012 Proceedings IEEE*. IEEE, 2012, pp. 2791–2795.