360BroadView: Viewer Management for Viewport Prediction in 360-Degree Video Live Broadcast

Qian Zhou, Zhe Yang, Hongpeng Guo, Beitong Tian and Klara Nahrstedt
Department of Computer Science, University of Illinois Urbana-Champaign
Champaign, Illinois, USA
{qianz,zheyang3,hg5,beitong2,klara}@illinois.edu

ABSTRACT

360-degree video is becoming an integral part of our content consumption through both video on demand and live broadcast services. However, live broadcast is still challenging due to the huge network bandwidth cost if all 360-degree views are delivered to a large viewer population over diverse networks. In this paper, we present 360BroadView, a viewer management approach to viewport prediction in 360-degree video live broadcast. We make some high-bandwidth network viewers be leading viewers to help the others (lagging viewers) predict viewports during 360-degree video viewing and save bandwidth. Our viewer management maintains the leading viewer population despite viewer churns during live broadcast, so that the system keeps functioning properly. Our evaluation shows that 360BroadView maintains the leading viewer population at a minimal yet necessary level for 97 percent of the time.

CCS CONCEPTS

• **Information systems** → **Multimedia information systems**.

KEYWORDS

360 video, viewer management, viewport prediction, live broadcast

ACM Reference Format:
Qian Zhou, Zhe Yang, Hongpeng Guo, Beitong Tian and Klara Nahrstedt. 2022. 360BroadView: Viewer Management for Viewport Prediction in 360-Degree Video Live Broadcast. In *ACM Multimedia Asia (MMAsia '22), December 13–16, 2022, Tokyo, Japan.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/3551626.3564939

1 INTRODUCTION

360° video has omnidirectional recording and makes the viewer have control of her viewing direction during viewing. Thus, it naturally fits with mixed reality applications in which tech giants (Apple, Microsoft, Google, Meta, etc.) are investing heavily, and is becoming an integral part of our content consumption through both video-on-demand (VOD) and live broadcast services.

Despite the bright prospect, it is unfeasible to deliver high-quality 360° content to a large viewer population over diverse networks. Viewport prediction (VP) is then proposed to save bandwidth, so that viewers with poor networks can also enjoy 360° video. The server splits a 360° video into small tiles, and encodes each tile into multiple copies of various bitrates. Ideally, only the tiles predicted to be in the viewer's future viewport will be transmitted with high bitrates; the other tiles will not be transmitted or transmitted with lower bitrates. In practice, a less accurate VP solution requests more extra tiles of high bitrates for fault tolerance in case the viewer's actual future viewport differs from the predicted one. Apparently, the more accurate VP we use, the more bandwidth we save.

Existing VP solutions fall into two categories. First, single-user viewport prediction (Single-VP) [16, 18, 19, 23] runs on the viewer's client-device, and uses the viewer's own historical viewport trajectory to predict her future viewport. Single-VP applies to both 360° VOD and live broadcast, but has high prediction accuracy only in the near future (< 1 second), after which it turns inaccurate and thus bandwidth-inefficient. Second, cross-user viewport prediction (Cross-VP) [14, 16, 17, 21, 25] runs on the server. It assumes that many viewers have watched the video, and their historical viewport trajectories were uploaded and stored on the server. Thus, when the current viewer is watching the same video, other viewers' historical viewports exist, and are used by the Cross-VP algorithm to predict the current viewer's future viewport. Cross-VP is more accurate and thus bandwidth-efficient in the distant future and should be used (alone or jointly with Single-VP) when available. However, due to its dependency on other viewers' historical viewports of the same video, Cross-VP normally applies to only 360° VOD, not live broadcast which has been watched by nobody.

In this paper, we seek to gain other viewers' historical viewports even during live broadcast, making existing Cross-VP solutions—which is more bandwidth-efficient than Single-VP but was applicable to only 360° VOD—now apply to live broadcast. We also minimize the number of Single-VP users and maximize the number of Cross-VP users throughout live broadcast despite viewer churns, further decreasing the overall network traffic.

First, we use a grouping strategy to make some live broadcast viewers use Single-VP and the others use Cross-VP. Based on research [2, 11, 23, 27], the playout among live broadcast viewers can be asynchronous for up to tens of seconds without affecting viewing experience. Thus, we let the server partition viewers into the *leading viewer group* and the *lagging viewer group*. When requesting video segments simultaneously, lagging viewers always get the content which was generated several (a constant) seconds earlier than leading viewers. Hence, each video segment is watched by leading viewers first and then by lagging viewers. Leading viewers use Single-VP because no historical viewports exist for the segments they will watch, and they keep uploading their actual viewports to the server; several seconds later when it is lagging viewers' turn to

MMAsia '22, December 13–16, 2022, Tokyo, Japan

ACM ISBN 978-1-4503-9478-9/22/12…$15.00
https://doi.org/10.1145/3551626.3564939

watch, the server has the historical viewports (from leading viewers) and conducts Cross-VP for lagging viewers. Thus, Cross-VP, which did not apply to live broadcast, now does.

Second, on base of the grouping strategy, we devise a viewer management approach to minimize the number of leading viewers (Single-VP users) and maximize the number of lagging viewers (Cross-VP users), further reducing the overall network traffic. Note that our viewer management is new: it serves 360° video viewport prediction, and should not be mistaken for the well-studied viewer management for multicast streaming in conventional 2D VOD systems [4]. Two challenges must be addressed:

1) How to reduce the leading viewer population to a small yet necessary level, and maintain it throughout live broadcast despite viewer churns? First of all, only high-bandwidth network (HBN) viewers, who have more bandwidth than Single-VP requires, are qualified to be leading viewers (Single-VP users). Instead of making all the HBN viewers be leading viewers, we only assign a necessary portion of them into the leading group, so that leading viewers are reduced but Cross-VP still gets enough inputs. The superfluous HBN viewers, together with all the low-bandwidth network (LBN) viewers, are assigned into the lagging group. Note that viewer churns are common in live broadcast, and anyone may quit watching at any time; even if there are enough leading viewers at the beginning, as they leave, at some point Cross-VP can no longer get enough inputs and will collapse. Our solution recycles the HBN viewers from the lagging group, and reassigns them as leading viewers to replenish the leading group. This process is not trivial: lagging viewers' watching progress is behind leading viewers', thus if they suddenly become leading viewers, they will experience a video forward jump and be disturbed. How to make reassignment *imperceptible* to them? We have a key observation that "elastic pieces" are widespread in videos and viewers are insensitive to their duration. Hence, our system conducts reassignment only during elastic pieces, keeping the viewers from noticing a video jump. *2) How to maintain the leading viewer population at a minimal yet necessary level?* As mentioned, leading viewers can leave at any time, but we cannot replenish the leading group until an elastic piece arrives. Our solution continuously optimizes and adjusts the leading group capacity by taking into account viewers' churn rate and elastic pieces' occurrence interval. Our contributions are:

- We propose 360BroadView, the first viewer management approach to grouping based viewport prediction in 360° video live broadcast. It minimizes the number of Single-VP users to reduce the overall network traffic.
- We devise solutions to maintain the leading viewer population at a minimal yet necessary level despite viewer churns, and in a way which is imperceptible to viewers.
- Our evaluation shows that: even in the harshest test case, 360BroadView succeeds in maintaining a minimal yet necessary leading viewer population for 97% of the time; its maintenance process is totally imperceptible to viewers.

2 BACKGROUND AND RELATED WORK

2.1 360° Video Tiled Streaming

360° video is bandwidth-demanding if entirely streamed. For efficient delivery, the server splits a 360° video into short segments, divides each segment into small tiles, and encodes each tile into multiple copies of various bitrates thus qualities. Viewport prediction (VP) algorithms (Section 2.2), running on the client-device and/or the server, predict the viewer's future viewport. The client-device conducts rate allocation [16, 18, 25, 26] and requests from the server higher-bitrate copies for the tiles which are predicted to be in the viewer's future viewport, while lower-bitrate or no copies for the tiles which are predicted to be out-of-sight.

2.2 Viewport Prediction

Single-VP. Single-user viewport prediction runs on the viewer's client-device. It uses the viewer's own historical viewport trajectory to predict her future viewport. Average prediction [19] (aka static [18], last sample [14]) uses the average of recent viewports as the predicted value of future viewport. Linear regression [19] models human head motion within a short time as linear, and conducts prediction based on this. Client navigation graph [16] treats each view-tile relation as a state, uses a viewer's past viewport trajectory to compute the transition probabilities between states, and predicts her future viewport and tiles. Feng *et al.* [6, 7] use a lightweight CNN which is trained and used at runtime for VP. The common drawback of a Single-VP solution is that it has high prediction accuracy only in the near future (< 1 second).

Cross-VP. Cross-user viewport prediction runs on the server. It assumes that many viewers have watched the same video, and their historical viewport trajectories were uploaded and stored on the server. Thus, for every segment the current viewer will watch, the viewports from other viewers are available, and taken as inputs to predict the current viewer's future viewport. Server navigation graph [16] is like a client one but trains the state machine using old viewers' trajectories. SEAWARE [17] is based on the navigation graph and brings in object detection as well as the view-object-tile relation as a state. CLS [25] clusters the old viewers with similar viewing behavior, classifies the current viewer into the cluster which her past behavior matches the best, and then predicts her viewports. Work [14] is like CLS but operated in spherical instead of Euclidean space. Flocking [21] presents a VP algorithm which gives a larger weight to an old viewer's viewports when using them to compute the current viewer's viewports, if the old and current viewers have similar historical viewports.

Single-VP is less accurate than Cross-VP, especially in the distant future [16, 17, 21, 25]. Thus, it needs to request more extra tiles for fault tolerance in case the viewer's actual future viewport differs from the predicted one, and costs more bandwidth than Cross-VP to achieve the same viewing quality. However, Cross-VP normally does not apply to live broadcast, because a live video is a premiere and other viewers' historical viewports do not exist.

3 OVERVIEW OF 360BROADVIEW

3.1 Offline Configuration

360BroadView involves: 1) 360° cameras streaming live 360° content to a server; 2) the server processing and broadcasting the live content; 3) viewers viewing the live content. 360BroadView is orthogonal to and used together with two existing VP algorithms (Single-VP and Cross-VP) and an existing video streaming approach [28]. A human administrator configures the server to

designate the exact VP algorithms to use. We select client navigation graph [16] as Single-VP and Flocking [21] as Cross-VP, but other VP solutions also work. We assume that two *constants* of the designated VP algorithms are provided:

- BW_{single}: the bandwidth cost of the designated Single-VP algorithm. A viewer can be a Single-VP user only if her available network bandwidth is $\geq BW_{single}$.
- N_{req}: the number of historical viewport trajectories required by the designated Cross-VP algorithm as inputs. Cross-VP cannot conduct accurate prediction with $< N_{req}$ inputs.

Each viewer, to become a 360BroadView user, must register for an account in advance, getting her viewer ID and the designated Single-VP algorithm installed on her client-device (e.g., headset). The designated Cross-VP algorithm is installed on the server.

3.2 Viewport Prediction & Tiled Streaming

As depicted in Fig. 1, the live broadcast cameras continuously record, upload and append 360° video segments to the video buffer on the server. The server divides each segment into L small tiles and encodes each tile into multiple copies of various bitrates.

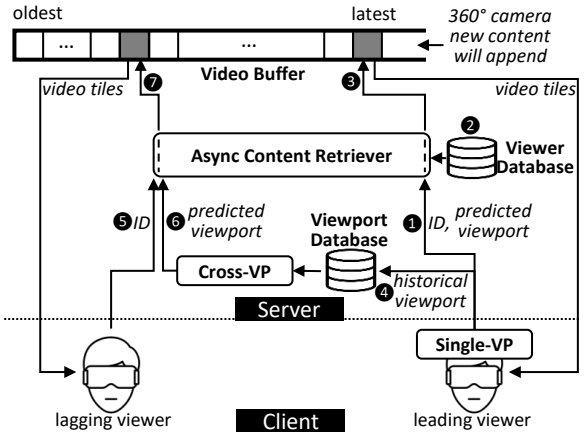


Figure 1: 360BroadView VP-based viewing.

In Fig. 1, each viewer is either a leading viewer or a lagging viewer; such a decision is made by 360BroadView's viewer management system (Section 3.3), then stored in the viewer database (❷) and told to the viewer. First, a leading viewer runs Single-VP on her client-device to get her *predicted viewport*. ❶ The client-device periodically sends to the server her ID and the *predicted viewport* (represented by a quaternion which describes the viewing direction, or a L-bit binary string where the ith ($1 \leq i \leq L$) bit value states whether the ith tile is in the future viewport or not). ❷ The Asynchronous Content Retriever finds her to be a leading viewer after viewer database lookup, ❸ then retrieves the video segment with timestamp $t_{leading}$ in the video buffer and delivers part of its tiles according to the *predicted viewport* to the viewer via an existing video streaming approach [28]. ❹ Besides, her actual viewports in the past few seconds are recorded by the client-device, and then uploaded and stored in the viewport database.

$t_{leading}$ denotes the generation timestamp of the video segment that leading viewers will receive, and is computed as below:

$$t_{leading} = t_{now} - T_{buf} \tag{1}$$

where t_{now} is the current timestamp; T_{buf} is the time interval between a segment's being generated by the camera and becoming available on the server for viewer retrieval, which consists of the time cost in video uploading, processing, and buffering for fluctuation elimination. E.g., at t_{now} = 10:00:15 AM, with T_{buf} = 10 seconds, the latest segment accessible to leading viewers was actually generated at 10:00:05 AM, and will be sent to them.

Second, ❺ a lagging viewer's client-device periodically sends the server her ID. ❷ The Asynchronous Content Retriever finds her to be a lagging viewer after viewer database lookup, and will retrieve the video segment with timestamp $t_{lagging}$ (older than $t_{leading}$) in the video buffer. Because the segment is old, leading viewers have watched it and uploaded their historical viewports. ❻ The server queries those viewports from the viewport database and feeds them into the Cross-VP algorithm, getting the *predicted viewport*. ❼ Then the tiles of segment $t_{lagging}$ and corresponding to the *predicted viewport* are retrieved and sent to the lagging viewer.

$t_{lagging}$ denotes the generation timestamp of the video segment that lagging viewers will receive, and is computed as below:

$$t_{lagging} = t_{leading} - T_{async} \tag{2}$$

where T_{async} is the two groups' watching progress difference enforced by the server. It is configured by the human administrator. We make it 5 seconds but other values are also fine.

As shown, when lagging and leading viewers request simultaneously, the former always get the content which was generated T_{async} earlier than the latter. In this way, leading viewers' watching progress is constantly T_{async} ahead, such that they can generate inputs for Cross-VP, whose outputs are used by lagging viewers.

3.3 Viewer Management

Once a live broadcast starts, a viewer can join in watching and leave at any time. As depicted in Fig. 2, when she joins, her client-device sends her ID and available network bandwidth BW_{viewer} to 360BroadView's viewer management system. She will be classified as a high-bandwidth network (HBN) viewer if $BW_{viewer} \geq BW_{single}$ (i.e., she can afford the bandwidth cost of Single-VP), otherwise as a low-bandwidth network (LBN) viewer. According to a viewer's bandwidth type, she will be assigned into either the leading group, which has a limited capacity N_{max} (to be determined in Section 5), or the lagging group, which is large enough to contain all viewers. Specifically, HBN viewers are assigned to the leading group when the leading viewer population is $< N_{max}$, otherwise to the lagging group. LBN viewers all go to the lagging group.

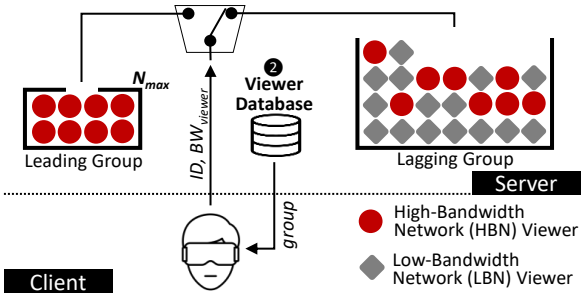


Figure 2: 360BroadView viewer management.

The viewer's information regarding bandwidth type ($type \in \{HBN, LBN\}$) and group ($group \in \{leading, lagging\}$) is stored in the viewer database (❷), in the format of $(ID, type, group)$. The viewer is then informed of her $group$. As is seen, due to our viewer management, there are at most N_{max} leading viewers (Single-VP users), and the others are all lagging viewers (Cross-VP users).

Viewer Maintenance. The leading group will have vacancies as some leading viewers leave, and it will be replenished in two ways: 1) when new HBN viewers join, they must go to those vacancies first; after the leading group is refilled (i.e., its population recovers to N_{max}), subsequent HBN viewers will join the lagging group. 2) under certain conditions, the HBN viewers in the lagging group will be recycled and reassigned to the leading group. We discuss the details of viewer maintenance in Section 4 and 5.

4 IMPERCEPTIBLE VIEWER MAINTENANCE

Viewer churns are common during live broadcast, and anyone may quit watching at any time; even if there are N_{req} (Section 3.1) or more leading viewers at the beginning, as they leave, at some point the leading viewer population will drop below N_{req} and Cross-VP can no longer get enough inputs to conduct accurate prediction.

Our solution recycles the HBN viewers from the lagging group and reassigns them as leading viewers to replenish the leading group. This process is not trivial: lagging viewers' watching progress is T_{async} (Section 3.2, 5 seconds) behind leading viewers', thus if they are reassigned, they will experience a T_{async} video forward jump and be disturbed. To make it *imperceptible*, our system conducts reassignment only when the viewer is watching an "elastic piece", whose duration can be adjusted without being noticed.

4.1 Elastic Piece

We notice that in live broadcast, key content (e.g., snippets of player competition, artist singing) is commonly interspersed with less important pieces (e.g., snippets of game break, audience reaction, penalty kick preparation, repeated slow-motion replay). We call the latter *elastic pieces* because decreasing their duration by several seconds is imperceptible to viewers (see user study in Section 6.2).

Fig. 3 shows the occurrences of elastic pieces in three representative live broadcast programs. In a tennis match, four game breaks happen within 30 minutes, starting at 2:02, 7:42, 21:01, 27:21. The corresponding video snippets are 24, 70, 67, 67 seconds long, and each is an elastic piece. It means that if the server only delivers, say, the first 19 seconds of the 24-second game break view to viewers and skips the remaining 5 seconds, viewers will not notice it.

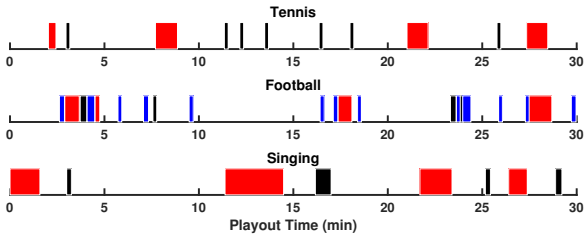


Figure 3: Elastic piece timelines. Red: break or preparation; Blue: repeated slow-motion replay; Black: audience reaction.

Elastic Piece Occurrence Interval. We use T_{ep} to denote the average interval between two adjacent elastic pieces. We find most programs like sports and singing shows have $T_{ep} < 10$ min.

Elastic Piece Recognition. We assume that the server knows which segments are elastic pieces, because a human videographer can easily recognize elastic pieces like audience reaction and quickly mark a newly generated video segment as elastic if it is, before it is uploaded to the server. An automatic solution based on machine learning is left to our future work.

4.2 Elastic Piece Based Reassignment

To imperceptibly reassign a HBN viewer from the lagging group to the leading group, we do not change the duration of the elastic piece for existing leading viewers while reducing it by T_{async} for the lagging viewer; after they finish watching their own versions of elastic pieces, their watching progresses will become the same.

To elaborate, let's assume that the leading group needs x more viewers to be refilled, and an elastic piece of audience reaction will be played to existing leading viewers during the time $[t_1, t_2]$. Originally it will be played to lagging viewers in $[t_1+T_{async}, t_2+T_{async}]$. Now under the control of 360BroadView, the broadcast server plays audience reaction to x randomly chosen HBN lagging viewers only during $[t_1+T_{async}, t_2]$, skips the remaining T_{async} elastic content, such that the x lagging viewers catch up with existing leading viewers and thus become leading viewers at t_2.

5 OPTIMAL VIEWER MAINTENANCE

We seek to minimize the number of leading viewers (Single-VP users) to reduce network traffic, on the premise that it is above N_{req} (Section 3.1), otherwise Cross-VP cannot get enough inputs. To this end, we must find an optimal value of the leading group capacity N_{max}. As mentioned in Section 4, leading viewers can leave at any time, but the leading group cannot be replenished until an elastic piece arrives. Thus, N_{max} must be larger than N_{req}: $\Delta = (N_{max} - N_{req})$ extra leading viewers are accommodated so that even if some leading viewers leave, the population will not go below N_{req} before next elastic piece arrives. If Δ is small, it may not take long before Δ leading viewers leave, then Cross-VP no longer gets N_{req} inputs; If Δ is large, there are many unnecessary leading viewers, violating our goal of minimization.

Our solution is to make N_{max} dynamic and adaptive to viewers' churn rate (how many viewers join/leave per minute) and elastic pieces' occurrence interval T_{ep} (Section 4.1), maintaining the leading viewer population at a minimal value above N_{req}. As shown in Section 3.3, 360BroadView has a viewer database: every time a viewer joins, she is classified as HBN or LBN, and her record is added to the viewer database; besides, we use a soft-state strategy for viewer leaving detection—if a viewer has not requested any video segment within the past 1 minute, she is regarded as gone and her record is removed from the database. Thus, 360BroadView can obtain the population and churn rate for each type of viewers. The system updates $\Delta n(t)$ *once per minute*:

$$\Delta n(t) = n^{join}_{HBN}(t) - n^{leave}_{leading}(t) \tag{3}$$

where $n^{join}_{HBN}(t)$ denotes the number of HBN viewers who join in watching within 1 minute before the moment t; $n^{leave}_{leading}(t)$ denotes

the number of leading viewers who leave within 1 minute before the moment t. Apparently, a positive $\Delta n(t)$ means that in the 1 minute, the leading group can maintain or increase its population (if the group is not full); a negative $\Delta n(t)$ means a leading viewer population decrease. The measured $\Delta n(t)$ in each minute fluctuates, and we use an exponential moving average $\widetilde{\Delta n}(t)$ for smoothness:

$$\widetilde{\Delta n}(t) = \lceil (1-\alpha) \cdot \widetilde{\Delta n}(t-1) + \alpha \cdot \Delta n(t) \rceil \quad (4)$$

where α is the weight given to the most recent observation. A larger α makes $\widetilde{\Delta n}(t)$ react more agilely but less smoothly. We empirically set it as 0.3 to achieve a good balance.

1) $N_{max} = N_{small}$ upon stable population. $\widetilde{\Delta n}(t) \geq 0$ indicates a stage of stable viewer population. In this situation, we can simply set N_{max} as N_{small}—a fixed value slightly larger than N_{req}:

$$N_{small} = \lceil \eta \cdot N_{req} \rceil \quad (5)$$

where η is a coefficient above 1. We empirically set $\eta = 1.1$, so the leading group has $\lceil 0.1 N_{req} \rceil$ more viewers than required. That small margin is used to tolerate occasional small population loss.

2) $N_{max} = N_{large}$ upon falling population. $\widetilde{\Delta n}(t) < 0$ indicates a stage where more leading viewers leave than join. Upon such a downtrend, N_{max} is raised to N_{large} so more viewers can be added to the leading group (when next elastic piece arrives) to cope with the ongoing population decrease. How much should N_{large} be? An analogy to explain our intuition: if we want to minimize the gasoline in our car, every time we reach a gas station, we add only the necessary gasoline that barely supports us to reach next station, where we will again add only necessary gasoline. Similarly, to minimize the leading viewer population, every time an elastic piece arrives, we reassign only necessary viewers from the lagging group to the leading group, such that the leading viewer population will just decrease to N_{req} when next elastic piece arrives; then we again reassign only necessary viewers, and repeat it. Thus,

$$N_{large} = N_{req} + |\widetilde{\Delta n}(t)| \cdot T_{remaining} \quad (6)$$

where $T_{remaining}$ denotes the remaining time before next elastic piece arrives, so $|\widetilde{\Delta n}(t)| \cdot T_{remaining}$ is the estimated population loss before next elastic piece arrives. As these viewers leave, N_{large} viewers will become N_{req}, and then be replenished in the arrived elastic piece. Thus, the leading viewer population is maintained at a minimal value but always above N_{req}. We get $T_{remaining}$ as:

$$T_{remaining} = max(0,\ T_{ep} - T_{elapsed}) \quad (7)$$

where T_{ep} (Section 4.1) is the interval between two adjacent elastic pieces; $T_{elapsed}$ is the time elapsed since the last elastic piece. Both can be easily measured and obtained by the server.

More Analysis on N_{large}. N_{large} is adaptive, and increases with $|\widetilde{\Delta n}(t)|$ and T_{ep} according to (6) and (7). This is reasonable because a larger $|\widetilde{\Delta n}(t)|$ means a higher viewer leaving rate, and a larger T_{ep} means fewer replenishment chances. In either case, the leading group should increase its capacity to get more viewers added to the group during replenishment, such that the population will not drop below N_{req} before next replenishment chance arrives.

6 EVALUATION

Our evaluation consists of three parts. 1) We implement three Single-VP and two Cross-VP algorithms for grouping based VP test, and show that without 360BroadView's viewer management, the system cannot keep functioning. 2) We conduct user study and confirm that our elastic piece based viewer reassignment is imperceptible. 3) We evaluate 360BroadView's maintenance performance and find that even in the harshest test case it successfully maintains a minimal yet necessary leading viewer population 97% of the time.

6.1 Demonstration of 360BroadView's Necessity

6.1.1 Methodology. **Viewport Trajectory Dataset.** We have investigated many datasets [1, 3, 5, 8, 12, 13, 15, 20, 24], and decided to use [24] to conduct emulation of live broadcast, because it is the only one used by almost all existing works (Section 2.2).

Baseline & Metric. We implement three Single-VP algorithms (linear regression, average prediction, and client navigation graph) and two Cross-VP algorithms (server navigation graph denoted as NG cross, and Flocking). All are introduced in Section 2.2. Viewers keep leaving in real life; for clear demonstration, here we make one leading viewer leave every 2 seconds (real viewer population traces are used for test in Section 6.3). In the baseline cases, they run without the aid of our viewer management. Each VP algorithm predicts which tiles will appear in the viewer's future viewport, and we use $recall = \frac{\#\ true\ positives}{\#\ actual\ tiles}$ to measure performance.

6.1.2 Results. Fig. 4 shows the two Cross-VP algorithms' performances: without the aid of 360BroadView, both NG cross and Flocking's *recall* values decrease with time and at some point to 0, because they get fewer and fewer inputs as leading viewers leave.

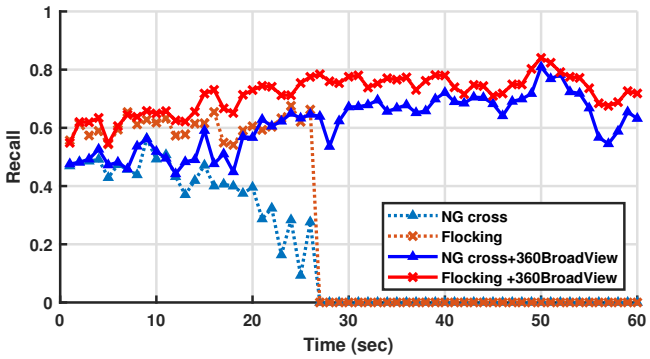


Figure 4: Recall of live broadcast VP wo/ & w/ 360BroadView.

With 360BroadView, the HBN viewers in the lagging group will be reassigned to the leading group if some leading viewers leave. And we see that both NG cross and Flocking now function properly from beginning to end. It shows that 360BroadView plays a vitally important role in grouping based live broadcast VP. We do not show the three Single-VP algorithms' performances because Single-VP is not impacted by other viewers' leaving.

6.2 User Study on Maintenance Imperceptibility

We test if viewers can notice any video jump in elastic pieces. We recruit 15 participants (13 males and 2 females) and make them each watch 3 videos (tennis, football and singing show). We have injected two 5-second video forward jumps to each video beforehand, by randomly picking two elastic pieces from each video and removing 5-second content from each elastic piece.

The user study contains two phases. 1) We did not tell the participants when each jump occurred and let them identify those moments in the videos. No one succeeded in finding any jump inserted by us. 2) Then we announced the moments and let the participants carefully observe them; we also displayed the raw videos to the participants as reference, and asked them if the jumps worsened viewing experience. No one said yes. The user study confirms that our elastic piece based viewer maintenance is imperceptible.

6.3 Evaluation on Maintenance Performance

6.3.1 Methodology. **Baseline & Metrics.** We compare two strategies which determine the leading group capacity N_{max}. The baseline uses a fixed capacity $\lceil 1.1N_{req} \rceil$; our solution uses an adaptive one (Section 5). The metrics of maintenance performance are:

(1) $\frac{N}{N_{req}}$: the ratio of leading viewers' actual population N to the required population N_{req}. It measures an approach's ability to maintain a minimal leading viewer population. Smaller $\frac{N}{N_{req}}$ is better because it means fewer leading viewers.

(2) $\tau = \frac{T(N<N_{req})}{T_{total}}$: the ratio of the time when $N < N_{req}$ to the total time of live broadcast. It measures an approach's ability to maintain a necessary leading viewer population. Smaller τ is better because $N < N_{req}$ means leading viewers are too few. E.g., in a 60-minute broadcast, if there are 6 minutes when N drops below N_{req}, then $\tau = 10\%$.

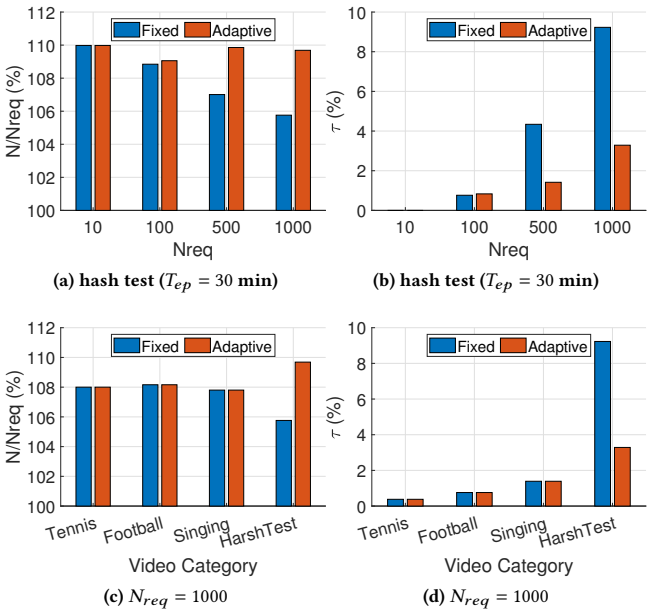
Test Conditions. We have various test conditions. We make N_{req} vary from 10 to 1000. As shown in Section 4.1, elastic pieces' interval T_{ep} is mostly < 10 min (for sports, singing shows, etc.); we test this condition, and also a harsh condition $T_{ep} = 30$ min. The combination of $N_{req} = 1000$ and $T_{ep} = 30$ min is the harshest.

Viewer Population Trace. We do not find any trace regarding how the number of 360° video live broadcast viewers varies with time; we argue that P2P live streaming viewer population traces are the best substitution. So we combine three P2P publications [9, 10, 22] and get the real 24-hour viewer population as the population of HBN viewers (LBN viewers are irrelevant to viewer maintenance); the data are self-repeated over days.

6.3.2 Results. **Impact of N_{req}.** Fig. 5a and 5b show the impact of N_{req}. When the fixed capacity is used, $\frac{N}{N_{req}}$ drops from 109.9% to 105.7% as N_{req} increases, because a larger N_{req} means more leading viewers need to be maintained, which is harder to achieve during a population decrease. τ increases from 0 to 9.2%, i.e., in the worst case, almost 10% of the time there are not enough leading viewers.

In contrast, when our adaptive capacity is used, $\frac{N}{N_{req}}$ is always close to 110% regardless of the N_{req} value; it is slightly higher than when the fixed capacity is used, because our solution dynamically enlarges the capacity to accommodate more leading viewers when detecting a population decrease. We see now τ increases with N_{req} very slowly, reaching only 3.2% when $N_{req} = 1000$. So, our solution maintains a necessary leading viewer population 97% of the time.

Impact of T_{ep}. Fig. 5c and 5d show the impact of T_{ep} on $\frac{N}{N_{req}}$ and τ. We see that $\frac{N}{N_{req}}$ is around 108% and τ is around 1% for the first three video categories regardless of using the fixed or adaptive capacity. This is because when T_{ep} is small (sports and singing shows have $T_{ep} < 10$ min), the leading group will have frequent, small replenishment. Thus, the adaptive capacity becomes a small value similar to the fixed capacity. The difference appears in the harsh test where T_{ep} is large (30 min): a large T_{ep} means that elastic pieces are infrequent and thus replenishment chances are fewer; the adaptive capacity becomes a large value to contain more viewers, such that the leading group can hold on till next replenishment chance arrives. In contrast, when the fixed capacity is used, the leading group cannot get enough viewers, resulting in a higher τ.



(a) hash test ($T_{ep} = 30$ min) (b) hash test ($T_{ep} = 30$ min)

(c) $N_{req} = 1000$ (d) $N_{req} = 1000$

Figure 5: Impact of N_{req} and T_{ep} on viewer maintenance.

To sum up, the adaptive capacity and the fixed capacity have remarkably different performances when N_{req} and T_{ep} are both large. Compared with the fixed capacity, the adaptive one uses 3.7% extra leading viewers but reduces τ by 64.3%, maintaining a necessary leading viewer population 97% of the time.

7 CONCLUSION

In this paper, we design and evaluate 360BroadView, the first viewer management approach to grouping based viewport prediction in 360° video live broadcast. Despite viewer churns in live broadcast, our viewer management maintains a necessary leading viewer population such that the system can keep functioning; also it minimizes the number of leading viewers to reduce network traffic. Our evaluation shows that 360BroadView maintains the leading viewer population at a minimal yet necessary level 97% of the time.

ACKNOWLEDGMENTS

This research was funded by the National Science Foundation CNS 1900875, by the Postdoctoral Fellowship Program at CS UIUC and by the Grainger College of Engineering funding. The presented views in the article are of the authors and do not represent the views of the funding organizations.

REFERENCES

[1] Jacob Chakareski, Ridvan Aksu, Viswanathan Swaminathan, and Michael Zink. 2021. Full UHD 360-Degree Video Dataset and Modeling of Rate-Distortion Characteristics and Head Movement Navigation. In *Proceedings of the 12th ACM Multimedia Systems Conference*. 267–273.

[2] Phil Cluff. 2019. The Low Latency Live Streaming Landscape in 2019. https://mux.com/blog/the-low-latency-live-streaming-landscape-in-2019. Online; accessed 7/1/2022.

[3] Xavier Corbillon, Francesca De Simone, and Gwendal Simon. 2017. 360-degree video head movement dataset. In *Proceedings of the 8th ACM on Multimedia Systems Conference*. 199–204.

[4] Asit Dan, Dinkar Sitaram, and Perwez Shahabuddin. 1996. Dynamic batching policies for an on-demand video server. *Multimedia systems* 4, 3 (1996), 112–121.

[5] Erwan J David, Jesús Gutiérrez, Antoine Coutrot, Matthieu Perreira Da Silva, and Patrick Le Callet. 2018. A dataset of head and eye movements for 360 videos. In *Proceedings of the 9th ACM Multimedia Systems Conference*. 432–437.

[6] Xianglong Feng, Zeyang Bao, and Sheng Wei. 2019. Exploring CNN-based viewport prediction for live virtual reality streaming. In *2019 IEEE International Conference on Artificial Intelligence and Virtual Reality (AIVR)*. IEEE, 183–1833.

[7] Xianglong Feng, Yao Liu, and Sheng Wei. 2020. LiveDeep: Online viewport prediction for live virtual reality streaming using lifelong deep learning. In *2020 IEEE Conference on Virtual Reality and 3D User Interfaces (VR)*. IEEE, 800–808.

[8] Stephan Fremerey, Ashutosh Singla, Kay Meseberg, and Alexander Raake. 2018. AVtrack360: an open dataset and software recording people's head rotations watching 360° videos on an HMD. In *Proceedings of the 9th ACM Multimedia Systems Conference*. 403–408.

[9] Xiaojun Hei, Chao Liang, Jian Liang, Yong Liu, and Keith W Ross. 2007. A measurement study of a large-scale P2P IPTV system. *IEEE transactions on multimedia* 9, 8 (2007), 1672–1687.

[10] Yan Huang, Tom ZJ Fu, Dah-Ming Chiu, John CS Lui, and Cheng Huang. 2008. Challenges, design and analysis of a large-scale p2p-vod system. *ACM SIGCOMM computer communication review* 38, 4 (2008), 375–388.

[11] Xing Liu, Bo Han, Feng Qian, and Matteo Varvello. 2019. LIME: understanding commercial 360 live video streaming services. In *Proceedings of the 10th ACM Multimedia Systems Conference*. 154–164.

[12] Wen-Chih Lo, Ching-Ling Fan, Jean Lee, Chun-Ying Huang, Kuan-Ta Chen, and Cheng-Hsin Hsu. 2017. 360 video viewing dataset in head-mounted virtual reality. In *Proceedings of the 8th ACM on Multimedia Systems Conference*. 211–216.

[13] Afshin Taghavi Nasrabadi, Aliehsan Samiei, Anahita Mahzari, Ryan P McMahan, Ravi Prakash, Mylène CQ Farias, and Marcelo M Carvalho. 2019. A taxonomy and dataset for 360 videos. In *Proceedings of the 10th ACM Multimedia Systems Conference*. 273–278.

[14] Afshin Taghavi Nasrabadi, Aliehsan Samiei, and Ravi Prakash. 2020. Viewport prediction for 360 videos: a clustering approach. In *Proceedings of the 30th ACM Workshop on Network and Operating Systems Support for Digital Audio and Video*. 34–39.

[15] Anh Nguyen and Zhisheng Yan. 2019. A saliency dataset for 360-degree videos. In *Proceedings of the 10th ACM Multimedia Systems Conference*. 279–284.

[16] Jounsup Park and Klara Nahrstedt. 2019. Navigation graph for tiled media streaming. In *Proceedings of the 27th ACM International Conference on Multimedia*. 447–455.

[17] Jounsup Park, Mingyuan Wu, Kuan-Ying Lee, Bo Chen, Klara Nahrstedt, Michael Zink, and Ramesh Sitaraman. 2020. Seaware: Semantic aware view prediction system for 360-degree video streaming. In *2020 IEEE International Symposium on Multimedia (ISM)*. IEEE, 57–64.

[18] Feng Qian, Bo Han, Qingyang Xiao, and Vijay Gopalakrishnan. 2018. Flare: Practical viewport-adaptive 360-degree video streaming for mobile devices. In *Proceedings of the 24th Annual International Conference on Mobile Computing and Networking*. 99–114.

[19] Feng Qian, Lusheng Ji, Bo Han, and Vijay Gopalakrishnan. 2016. Optimizing 360 video delivery over cellular networks. In *Proceedings of the 5th Workshop on All Things Cellular: Operations, Applications and Challenges*. 1–6.

[20] Yashas Rai, Jesús Gutiérrez, and Patrick Le Callet. 2017. A dataset of head and eye movements for 360 degree images. In *Proceedings of the 8th ACM on Multimedia Systems Conference*. 205–210.

[21] Liyang Sun, Yixiang Mao, Tongyu Zong, Yong Liu, and Yao Wang. 2020. Flocking-based live streaming of 360-degree video. In *Proceedings of the 11th ACM Multimedia Systems Conference*. 26–37.

[22] Long Vu, Indranil Gupta, Klara Nahrstedt, and Jin Liang. 2010. Understanding overlay characteristics of a large-scale peer-to-peer IPTV system. *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)* 6, 4 (2010), 1–24.

[23] Shuoqian Wang, Xiaoyang Zhang, Mengbai Xiao, Kenneth Chiu, and Yao Liu. 2020. SphericRTC: A System for Content-Adaptive Real-Time 360-Degree Video Communication. In *Proceedings of the 28th ACM International Conference on Multimedia*. 3595–3603.

[24] Chenglei Wu, Zhihao Tan, Zhi Wang, and Shiqiang Yang. 2017. A dataset for exploring user behaviors in VR spherical video streaming. In *Proceedings of the 8th ACM on Multimedia Systems Conference*. 193–198.

[25] Lan Xie, Xinggong Zhang, and Zongming Guo. 2018. Cls: A cross-user learning based system for improving qoe in 360-degree video adaptive streaming. In *Proceedings of the 26th ACM international conference on Multimedia*. 564–572.

[26] Praveen Kumar Yadav and Wei Tsang Ooi. 2020. Tile rate allocation for 360-degree tiled adaptive video streaming. In *Proceedings of the 28th ACM International Conference on Multimedia*. 3724–3733.

[27] Jun Yi, Shiqing Luo, and Zhisheng Yan. 2019. A measurement study of YouTube 360 live video streaming. In *Proceedings of the 29th ACM Workshop on Network and Operating Systems Support for Digital Audio and Video*. 49–54.

[28] Lei Zhang, Yanyan Suo, Ximing Wu, Feng Wang, Yuchi Chen, Laizhong Cui, Jiangchuan Liu, and Zhong Ming. 2021. TBRA: Tiling and Bitrate Adaptation for Mobile 360-Degree Video Streaming. In *Proceedings of the 29th ACM International Conference on Multimedia*. 4007–4015.