Prioritized Buffer Control in Two-tier 360 Video Streaming

Fanyi Duanmu, Eymen Kurdoglu, S. Amir Hosseini, Yong Liu and Yao Wang
New York University, Tandon School of Engineering
Brooklyn, New York 11201, USA

ABSTRACT

360 degree video compression and streaming is one of the key components of Virtual Reality (VR) applications. In 360 video streaming, a user may freely navigate through the captured 3D environment by changing her desired viewing direction. Only a small portion of the entire 360 degree video is watched at any time. Streaming the entire 360 degree raw video is therefore unnecessary and bandwidth-consuming. One the other hand, only streaming the video in the predicted user's view direction will introduce streaming discontinuity whenever the the prediction is wrong. In this work, a two-tier 360 video streaming framework with prioritized buffer control is proposed to effectively accommodate the dynamics in both network bandwidth and viewing direction. Through simulations driven by real network bandwidth and viewing direction traces, we demonstrate that the proposed framework can significantly outperform the conventional 360 video streaming solutions.

CCS CONCEPTS

• **Networks** → **Mobile networks**; *Application layer protocols*; • **Computing methodologies** → *Virtual reality*;

KEYWORDS

360 Degree Video, Virtual Reality, Video Streaming, Buffer Control

ACM Reference format:
Fanyi Duanmu, Eymen Kurdoglu, S. Amir Hosseini, Yong Liu and Yao Wang. 2017. Prioritized Buffer Control in Two-tier 360 Video Streaming. In *Proceedings of VR/AR Network '17, Los Angeles, CA, USA, August 25, 2017,* 6 pages.
https://doi.org/10.1145/3097895.3097898

1 INTRODUCTION

In recent years, Virtual Reality (VR) technologies have been rapidly commercialized. A variety of applications have been developed continuously to meet the market demands and consumer expectations, such as immersive cinema, gaming, education/training, tele-presence, social media, and healthcare, etc. Therefore, the delivery of ultra high quality 360 degree video has become critically important for the wide adoption of VR. The main differentiator is to provide end consumers with omni-directional viewing flexibility and immersive video experience. Some preliminary 360 video services are now available on several major video platforms, such as YouTube, Facebook, etc. However, the current delivery solutions treat 360 videos as regular videos and stream the entire 360 degree view scope to users regardless of their view directions. Compared with the traditional video streaming, 360 degree video streaming confronts unique new challenges. Firstly, to deliver an immersive VR experience, 360 video has much higher bandwidth requirement. For example, a premium quality 360 video with 60 frames per second, 4K resolution can consume bandwidth up to multiple Gigabits-per-second (Gbps). Secondly, user view direction dynamics is a new dimension of freedom in 360 degree video streaming. A user may arbitrarily change or navigate her viewing direction and expect to see the scene in the new viewing direction immediately. In recent years, several solutions have been proposed to address 360 video streaming. We summarize them into three categories:

Category 1: 360 Video Source Representation. Videos captured from different view angles first need to be projected to a 2D plane before further processing. Facebook proposed the cube-map [7] and pyramid [8] projection methods and encoding schemes in 2016, to specifically address on-demand 360 video streaming, with 25% and 80% compression improvements reported, respectively. The Joint Video Exploration Team (JVET) also proposed several projection solutions, including Icosahedral projection (ISP) [15], Segmented Sphere Projection (SSP) [14], Truncated Square Pyramid Projection (TSP) [4], Octahedron Projection (OHP) [9], etc.

Category 2: Source Bit Allocation. Different view regions have different perceptual quality implications, consequently deserve different numbers of coding bits. In [3], a region-adaptive smoothing scheme is proposed to reduce the bitrate spent within the polar regions of equi-rectangular 360 videos through Gaussian filtering. A 20% bitrate reduction is reported with unnoticeable perceptual quality degradation.

Category 3: View-based Streaming. In [13], a few tile-based encoding and streaming solutions are proposed, including scalable coding scheme and simulcast coding scheme. Video tiles that cover the whole 360 scene are coded in multiple rates. Depending on the Field of View (FOV), tiles within or close to the predicted FOV are fetched with higher bitrate while tiles far away from the predicted FOV are fetched with lower bitrate. In [10], a view prediction based framework is proposed by only fetching the video portions desirable to the end user to reduce the bandwidth consumption. A dynamic video chunk adaptation scheme is implemented to adjust tile coverage based on the view prediction accuracy. An estimated 80% maximum rate reduction is reported without considering the coding efficiency loss due to video tiling and bandwidth variations.

Inspired by the previous work [5], we propose a novel two-tier dynamic 360 video streaming framework, which encodes 360 video into two tiers and adaptively streams the two tiers of the video to

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.
VR/AR Network '17, August 25, 2017, Los Angeles, CA, USA
© 2017 Association for Computing Machinery.
ACM ISBN 978-1-4503-5055-6/17/08…$15.00
https://doi.org/10.1145/3097895.3097898

cope with the variations in the network bandwidth and user view direction changes. Unlike most previous solutions, our proposed two-tier framework is able to deal with the unexpected viewing direction changes and network variations simultaneously. Furthermore, our proposed solution is source-representation-independent. Any above-mentioned projection methods (such as Cube-map, Icosahedral, etc.) and source bit-allocation approach (e.g., region-adaptive smoothing) can be easily incorporated into our framework. This work significantly extends our previous work [5] from three perspectives. Firstly, the target buffer length in this work is flexible and can be jointly optimized based on network statistics and view prediction accuracy. Secondly, the view prediction is improved with a linear model, which we prove to be more accurate than the "sample-and-hold" model in our previous work. Finally, we introduce a systematic formulation for dynamic scheduling of 360 video based on prioritized buffer control to better utilize available bandwidth and enhance user quality of experience.

The rest of this paper is structured as follows. In Section 2, our proposed two-tier framework is illustrated and the design principles are discussed. In Section 3, we formulate the prioritized buffer-based 360 video scheduling process in our system as a Proportional-Integral (PI) control problem and propose our optimization methodologies. In Section 4, we describe our experimental settings in details against two benchmark configurations. In Section 5, the simulation results are provided to demonstrate the potentials of the proposed solution. Section 6 concludes this paper with future directions.

2 OVERVIEW OF TWO-TIER 360 VIDEO STREAMING

As illustrated in Figure 1, a 360 video is partitioned into non-overlapping time segments, and each segment is encoded into a base-tier (BT) chunk and multiple enhancement-tier (ET) chunks. A BT chunk encodes the entire 360 view span at a low bitrate to provide basic quality. BT chunks for future time segments are prefetched in a long display buffer to cope with network bandwidth variations and guarantee that any desired FOV can be rendered with minimum stalls at the client. Each ET chunk encodes video within a view window with certain view coverage (VC) (e.g., 120 degrees) centered at certain direction. To provide quality differentiation, multiple ET chunks can be generated for the same view window, but at different coding rates. For complete coverage and smooth transition between view windows, the view windows of ET chunks in the same time segment are *overlapping* and cover the whole 360 view span. All the pre-coded chunks are stored in the streaming server. The client will decide and request a particular rate version from a particular tier, according to the predicted view direction for the segment, the predicted download bandwidth in the next request interval, and the buffer status of each tier.

In our current implementation, the BT chunks are encoded at a basic rate, which is expected to be sustainable even when the network bandwidth is low. Therefore, the bandwidth wasted on covering scenes outside of the user FOV is limited. Since a BT chunk encodes the entire 360 degree scene, it is always useful for rendering no matter how dynamically a user changes her view direction during the streaming session. In the extreme case of aggressive prefetching, as long as the average network bandwidth is above the

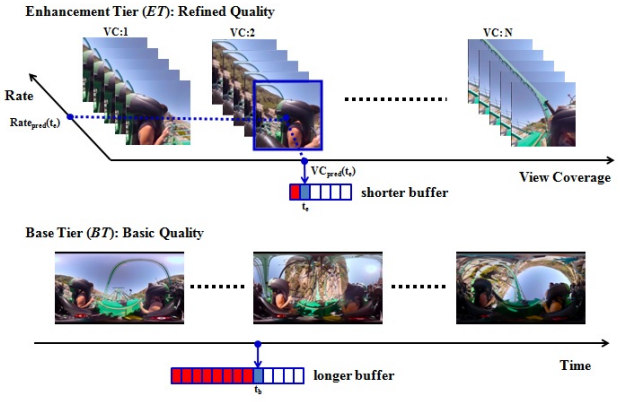

Figure 1: Two-tier 360 Video Streaming System.

average coding rate of the BT, the end user is guaranteed to receive a continuous 360 video experience with basic quality, regardless of how instantaneous network bandwidth varies, and regardless of how abruptly/frequently the user changes her view direction.

The ET chunks are designed to improve the streamed video quality whenever there is additional bandwidth available after the base tier chunks are delivered. Since both future available bandwidth and user viewing direction are generally unknown, ET chunks will be prefetched in an *opportunistic* fashion, with the help of available bandwidth estimation and view direction prediction. Specifically, at time t, when additional bandwidth is available, one can prefetch an ET chunk for $t + \Delta$ covering the predicted view window at that time. As studied in [10], the long-term head motion prediction is very difficult. Therefore, the ET chunks in our system are prefetched in a relatively shallow buffer (e.g., up to 5 second ahead) so that the delivered view window mostly coincides with the actual user FOV. When user view directions are predicted accurately and ET chunks are received successfully, the client video player can combine the ET chunk with the prefetched BT chunk for an enhanced quality. Even when the view prediction fails (e.g., due to unexpected head motion) or when the requested ET chunk does not arrive promptly before its deadline (e.g., due to sudden bandwidth decrease), the client can still render the desired view with basic quality from the prefetched BT.

3 PRIORITIZED BUFFER-BASED 360 VIDEO STREAMING

We formulate two-tier 360 video streaming as a dynamic scheduling problem. Similar to the Dynamic Adaptive Streaming over HTTP (DASH) [11] framework, we consider the scheduling as a discrete time process. At each time slot, a client strategically prefetches video chunks from both tiers based on view direction and available bandwidth predictions. The goal is to maximize the rendered video quality of the streaming session, while both the network bandwidth and user view direction may vary over time. Beyond our previous preliminary two-tier system study [5], in this work, we propose to set up strict priority between multiple design objectives and develop scheduling algorithms to achieve the desired priority. In our current

design, we believe that playback continuity is the most important and therefore we give the highest priority to download the base tier chunks in near future so that we can always render a basic-quality version even if the high-quality version is not available at the desired FOV, either due to view prediction error or enhancement-tier buffer underflow. Similar to many DASH work, we measure the buffer length using the buffered video time. If the current base-tier buffer length is less than the target buffer length q^b_{ref}, one should always sequentially download the BT chunks until the BT buffer reaches q^b_{ref}.

After downloading enough BT chunks, the residual bandwidth will be used to download the ET chunks. In this paper, we formulate the ET chunk scheduling problem as a buffer-based feedback control problem, leveraging on our previous study for buffer-based DASH [12]. Let $q^e(t)$ be the buffered video time for enhancement tier at time t. When a chunk k with future playback time is prefetched, the evolution of $q^e(t)$ can be approximated by the fluid model provided in Eq. (1), where $\mathbf{1}(\cdot)$ is the indicator function, and $\bar{b}(k)$ is the average bandwidth when downloading chunk k, τ is the video duration for each chunk, $s^e(k)$ is the size of chunk k, and $t^{(s)}_k$, $t^{(f)}_k$ is the starting and finishing time of downloading chunk k.

$$\frac{d}{dt}q^e(t) = \frac{\bar{b}(k)\tau}{s^e(k)} - \mathbf{1}(q^e(t) > 0), \quad t \in (t^{(s)}_k, t^{(f)}_k] \tag{1}$$

One can select the request rate version of an ET chunk by setting up a target buffer length q^e_{ref} for the ET. If the current buffer length is less than q^e_{ref}, one should be conservative and choose a chunk with size $s^e(k) < \hat{b}(k)\tau$, the estimated bandwidth budget, such that more video time can be accumulated; if the current buffer length is greater than q^e_{ref}, one can be more aggressive and choose a chunk with size $s^e(k) > \hat{b}(k)\tau$, so that the accumulated video time can be reduced to q^e_{ref}. As shown in [12], traditional feedback control algorithms, such as Proportional-Integral (PI) controllers, can maintain the target buffer length very well. The target rate $\hat{R}(k)$ of ET chunk k can be determined based on buffer evolution as

$$u(k) = K_P(q^e(t^{(s)}_k) - q^e_{ref}) + K_I \sum_{t=0}^{t^{(s)}_k}(q^e(t) - q^e_{ref}), \tag{2}$$

$$\hat{R}(k) = \frac{s^e(k)}{\tau} = min\left[(u(k)+1), \frac{\Delta(k)}{\tau}\right] \cdot \hat{b}(k), \tag{3}$$

where K_P and K_I are the proportional and integration gain control coefficients, respectively, $u(k)$ is the control signal, and $\Delta(k)$ is the remaining time till the display deadline of ET chunk k.

The target buffer length tuple $\langle q^b_{ref}, q^e_{ref}\rangle$ for base and enhancement tiers reflects the trade-off between robustness and quality. A large q^b_{ref} achieves high robustness against variations in both network bandwidth and user view direction changes, but at the cost of reduced likelihood to download the enhancement tier chunks, lowering the rendered video quality. A large q^e_{ref} also achieves high robustness against network bandwidth variation, but is *vulnerable* to user view direction changes, simply because it is more difficult to predict user's view direction into the far future, and a prefetched enhancement-tier chunk is useless if its view coverage does not cover the user's actual FOV for that segment. In this study, we encode the BT chunks at a basic rate expected to be sustainable even at low bandwidths. The bitrates of the ET chunks, on the other hand, are more significant compared to the BT. Target buffer length selection for ET chunks is a more interesting and important challenge in our two-tier streaming framework. Therefore, we present a formulation to determine the target enhancement tier buffer length q^e_{ref}. Ideally, we would like to maximize the rendered video quality. However, since the video quality is generally monotonically increasing with the average *video rendering rate* (in terms of bits per viewing area), we try to maximize the delivered video rate instead. This design obviates the prior dependency knowledge between the video quality and video rate, which is typically content dependent. Besides, it also leads to a simpler solution, because our design parameter (i.e., q^e_{ref}) directly impacts the rate.

Because the base-tier buffer length in our system is long, we assume that the base-tier chunks are mostly delivered in time for display, so that for each video segment, we either receive only the base-tier or both the base-tier and the enhancement-tier chunks. The base-tier chunks are coded to cover the entire area of 360 video with the total rate of R_b (in bits/second) and therefore the video rendering rate is R_b/A_b, where A_b is the viewing area of the 360 video. Let $\bar{R_e}$ and A_e denote the average enhancement-tier rate and the coverage area of each ET chunk, respectively. Since that the predicted view direction for a delivered video segment may not be the same as the actual user viewing direction, therefore, not all received chunks for the enhancement-tier are useful. In general, only a portion of each decoded frame in the delivered chunk may overlap with the user's FOV for that frame. Here we introduce α to denote the average ET View Prediction Accuracy (VPA), namely the average overlapping ratio between the predicted view coverage and user's actual FOV, and γ to denote the average ET Chunk Pass Rate (CPR), namely the likelihood that a requested ET chunk can be delivered successfully before its display deadline. Therefore, the expected Video Rendering Rate (VRR) can be expressed as

$$\begin{aligned} R_{VRR}(q^e_{ref}) &= \gamma(\alpha(\frac{R_b}{A_b} + \frac{\bar{R_e}}{A_e}) + (1-\alpha)\frac{R_b}{A_b}) + (1-\gamma)\frac{R_b}{A_b} \\ &= \frac{R_b}{A_b} + \alpha\gamma\frac{\bar{R_e}}{A_e}, \end{aligned} \tag{4}$$

where α and γ are both functions of q^e_{ref}. Intuitively, α decreases as q^e_{ref} increases because the view prediction into far future becomes less reliable. γ increases as q^e_{ref} increases because a longer ET buffer is more likely to absorb the temporary mismatch between the real network throughput and predicted bandwidth. Obviously, there is an intrinsic trade-off between increasing $\alpha(q^e_{ref})$ (i.e., view prediction accuracy) and increasing $\gamma(q^e_{ref})$ (i.e., ET chunk pass rate) when selecting q^e_{ref}. Assuming that one can predict the bandwidth for each chunk accurately, and that there are sufficient rate versions for the same time segment to match the available bandwidth, then we can approximate $\bar{R_e} = \bar{T} - R_b$, where $\bar{T}$ is the average available bandwidth. Eq. (4) implies that we should select the q^e_{ref} that maximizes the product of $\alpha(q^e_{ref})\gamma(q^e_{ref})$ to maximize the delivered video rate. In our implementation, we determine $\alpha(q^e_{ref})$

and $\gamma(q^e_{ref})$ experimentally by simulating our system using different q^e_{ref} with a variety of network traces and view traces. Please note that γ is determined by both network statistics and also video rate versions provided on the streaming server. In our system, four different rate versions are provided and used for simulations. α is mainly determined by view prediction methodologies and the number and size of VCs in the enhancement-tier.

4 EXPERIMENTAL SETTINGS

4.1 Video Preparation

We downloaded two sample 360 videos[1] (1920x3840, 30Hz) from YouTube, as shown in Figure 2. For simplicity, we neglect the possible rate fluctuations caused by the content variation and assume the video rate control is perfect such that each ET video chunk is coded with constant rate. Horizontally the 360 video is divided into twelve View Coverages (VC). Each VC spans 120 degrees with 30 degree stride. Vertically the video is divided into three VCs. Each VC spans 90 degrees with a 45 degree stride. Therefore, for each BT chunk, there are totally 36 corresponding VCs in the ET to cover different viewing directions. For simplicity, we assume these independently-decodable video chunks are coded with a fixed group of picture (GOP) length (i.e., 1 second). We also assume that the user FOV has the same dimension as our VCs (i.e., 120° × 90°).

Figure 2: Sample Frames in Test Videos "MegaCoaster", "Amsterdam" downloaded from YouTube

4.2 Network Bandwidth Traces

To simulate dynamic networks with significant bandwidth variations, we use the traces collected over a 3.5G HSPA cellular network using the methodologies described in [6]. Sample traces are illustrated in Figure 3. These traces represent the most typical bandwidth variations in a cellular network. We further scale up the original bandwidth sample values to adapt to the 4K/30Hz 360 degree video bitrate range (i.e., up to 400 Mbps).

4.3 View Direction Traces

We collected the view direction traces from four users as follows. Each user wears a Google Cardboard [2] with a Motorola Nexus-6 smart-phone playing our test 360 videos. Simultaneously, a head tracker [1] equipped on Cardboard dynamically transmits motion data (e.g., yaw, pitch, roll, etc.) to a nearby PC for data recording. Figure 4 illustrates a sample trace generated by concatenating the view direction data from 4 users over the "RollerCoaster" test sequence.

[1]Sample videos download links: https://www.youtube.com/watch?v=-xNN-bJQ4vI and https://www.youtube.com/watch?v=FzrkpXlRP1M.

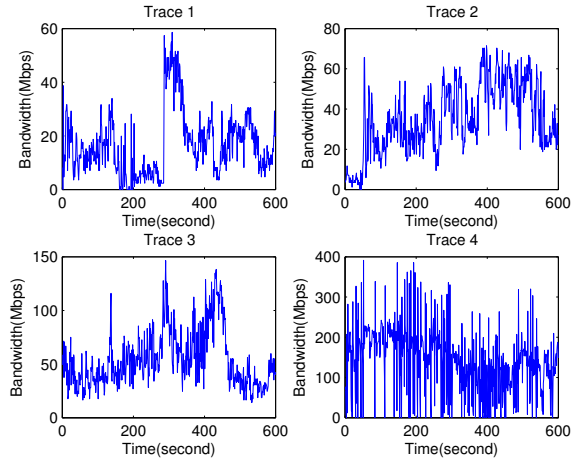

Figure 3: Sample Network Bandwidth Traces after Scaling.

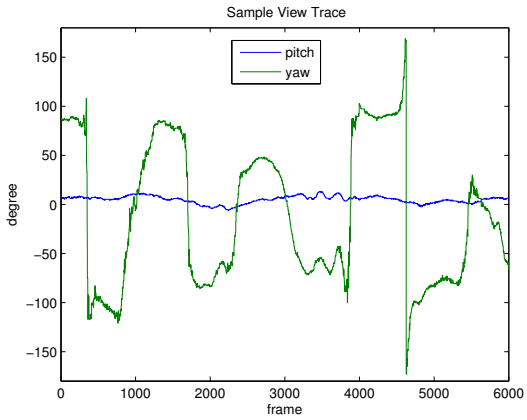

Figure 4: Sample view trace in yaw and pitch directions. On yaw trace, the -180 degree coincides with +180 degree.

4.4 Proposed Two-Tier Solution

In our proposed Two-Tier system (TTS), the BT stores the 360 degree view segments (each lasting 1 second) coded at a low bitrate R_0 =10 Mbps to provide basic quality. The ET stores all possible VCs (120° × 90°). The VCs are coded at three different rates (R_1, R_2 or R_3). The initial buffer length for BT is set to 20 seconds. Upon complete reception of a chunk, the client estimates the available bandwidth for the next chunk to be equal to the average throughput for downloading the last chunk (from either BT or ET), i.e., $\hat{b}(k+1) = s(k)/T(k)$ where $s(k)$ is the size of the last received chunk k and $T(k)$ is the transmission time of chunk k. We select the target video rates based on the network bandwidth cumulative distribution function (CDF) in our traces. Specifically, R_0+R_1, R_0+R_2, R_0+R_3 are chosen at 40%, 60% and 80% percentile of the network bandwidth CDF, respectively. Furthermore, the client predicts the viewing direction for segment $n+1$ through linear regression based on the past 30 view samples and correspondingly determine the VC to be

fetched. When the BT queue length is less than q^b_{ref}, the client will always sequentially download the BT chunks. When the BT queue length is above q^b_{ref} (i.e., 10 second), the client will download ET chunks with target rate $\hat{R}(n+1)$ according to Eq. (2) and Eq. (3). Let $R_{n+1,1}$, $R_{n+1,2}$ and $R_{n+1,3}$ denote the actual rates of the low-quality, medium-quality and the high-quality versions of incoming chunk $n+1$ for the target VC, respectively. If $\hat{R}(n+1) \geq R_{n+1,3}$, the client will request the high-rate ET chunk. If $\hat{R}(n+1) < R_{n+1,3}$ and $\hat{R}(n+1) \geq R_{n+1,2}$, the client will request the medium-rate ET chunk. Otherwise, the client will request the low-rate ET chunk.

We use the received video rendering rate to quantify the system performance. Let $R^b(n)$ indicate the rate of the BT chunk for time segement n and $R^e(n)$ denote the rate of the ET chunk for a particular VC for time segment n. $R^e(n) = 0$ if the ET chunk is not available at the display time. We define w_f as the overlapping portion between the ground-truth FOV per frame (obtained from the view trace) and the VC of the downloaded ET chunk, w_b the overlapping portion of the desired FOV and the 360 view decoded from the BT. Therefore, the VRR in Segment n is defined as $VRR(n) = w_b R^b(n) + w_f R^e(n)$. Here we assume that the rendered view covers $120^\circ \times 90^\circ$ so w_b=1/6. In the rare case when the BT buffer is empty at the display time, then $VRR(n) = 0$.

4.5 Benchmark Solution 1 (*BS1*): Full-360 Streaming

BS1 simulates the typical *DASH* streaming framework for 360 videos, in which the entire equi-rectangular videos are pre-encoded using multiple rates. For a fair comparison, we select the same rate setting as our proposed TTS (i.e., R_B=R_0=10 Mbps, R_L=R_0+R_1, R_M=R_0+R_2 and R_H=R_0+R_3). The initial buffer length is configured the same as TTS (i.e., 20 seconds) and the target buffer length is also 10 second. Similarly to TTS, the client estimates the sustainable transmission rate for the incoming segment $n+1$ to be equal to the measured throughput for downloading the last received chunk n, and then accordingly chooses a rate to request over the next segment $n+1$ using *PI*-controller. The VRR for the desired view over each segment is therefore the rate of the downloaded chunk scaled by $w_b = 1/6$ and is 0 when the display buffer is empty.

4.6 Benchmark Solution 2 (*BS2*): VC-Streaming

In *BS2*, only VCs are pre-coded and stored on the server. Each VC covers $120^\circ \times 90^\circ$ view scope similarly as our TTS. Each ET chunk is encoded directly with four rates consistent with our proposed TTS (i.e., R_B=R_0=10 Mbps, R_L=R_0+R_1, R_M=R_0+R_2 and R_H=R_0+R_3). Similar to TTS, at time n, it predicts the view direction at $n+1$ using linear interpolation based on the past 30 samples. If the requested segment does not arrive completely before the display deadline, then we set VRR = 0 over that second. Otherwise, we use the portion of the downloaded VC that overlaps with the user desired FOV to calculate its VRR as $w_f \cdot R$, where $R \in \{R_B, R_L, R_M, R_H\}$. To be fair, we apply the same PI controller and the initial enhancement tier buffer length (i.e., 1 second) as in TTS configuration.

5 SIMULATION RESULTS

Our proposed TTS is simulated and compared with the two benchmark solutions. We evaluate the performance directly using the delivered Video Rendering Rate (i.e, VRR) and Video Freeze Ratio (VFR). The delivered VRR is defined as the received bits per rendered area, averaged over all displayed frames. The VFR is the percentage of total time that video buffer underflows (i.e. no bits are available for the user FOV at the display time). Four different network traces (each of 600 seconds) and two view traces are used for simulation. For simplicity, the view traces are played in loops for a total duration of 600 seconds. The parameters of *PI*-controller (i.e., K_P and K_I) are chosen through an exhaustive search at each q^e_{ref} over the concatenated network trace, to maximize the ET chunk pass rate, where K_P starts from 0.5 up to 1.0 with a stride of 0.1 and K_I starts from 0 up to 0.20 with a stride of 0.01.

5.1 Enhancement Tier Target Buffer Length Optimization

The q^e_{ref} in our system is optimized off-line over the collected view and network traces. The optimal operation point is jointly determined by the average VPA (i.e., α) and the average CPR (i.e., γ). To illustrate, the VPA over our concatenated view trace is plotted in Figure 5 (red curve). Specifically, we fit the past 30 samples using a linear model, i.e., $y = At + b$, where t are sample timestamps and y are the corresponding viewing angles. The coefficients (i.e., A and b) are derived to minimize the sum of the least-square error between the true and predicted user viewing angles. Then we apply the derived linear model to predict the viewing angle for the incoming ET chunk at $t+\delta$, i.e., $\hat{y} = A(t+\delta) + b$, where δ is positive. The average CPR under our combined network trace is provided in Figure 5 (blue curve). As shown from Figure 5, the optimal operation point is $q^e_{ref} = 1$, corresponding to the peak on the purple curve in Figure 5. The simulation results over different operation points of q^e_{ref} are provided in Table 1 for comparison. This result coincides with our conjecture that the q^e_{ref} that maximizes the product of α and γ also maximizes the VRR.

5.2 Performance Comparison with Benchmark Solutions

The delivered VRR and video freeze ratio (VFR) using three solutions are presented in Table 2. The bandwidth traces are visualized in Figure 3 for reference.

In *BS1*, the long buffer setting effectively absorbs the network bandwidth variations and the available bandwidth is well-utilized. However, due to the ignorance of user FOV in streaming, the VRR is only 1/6 of the encoded rectangular video rate.

In *BS2*, when bandwidth is sufficient, the high-rate chunks can be successfully delivered. The delivered VRR is maximized when the view prediction is also accurate (particularly when the user viewing direction is relatively stable or changing smoothly). However, when the bandwidth suddenly decreases, the shallow enhancement-tier buffer may occasionally underflow, resulting in annoying frequent video freezes and severely degrade the user experience.

In our proposed TTS, the advantages of the two benchmark solutions are integrated. On one hand, the base-tier long buffer

Table 1: Delivered Video Rendering Rate at Different Target ET Buffer Lengths

q^e_{ref}	1-second	2-second	3-second	4-second	5-second
VRR ($Mbps$)	49.80	49.15	47.10	46.24	45.13

Table 2: Performance Comparisons: Average Video Rendering Rate (Mbps) / Video Freeze Ratio (%)

Network Trace Solution	1 *BS*1	1 *BS*2	1 TTS	2 *BS*1	2 *BS*2	2 TTS	3 *BS*1	3 *BS*2	3 TTS	4 *BS*1	4 *BS*2	4 TTS
RollerCoaster	2.8/12%	10.8/27%	7.9/6%	5.9/4%	27.0/10%	21.1/4%	9.0/1%	40.2/2%	36.6/0%	23.0/0%	93.1/8%	108.0/0%
Amsterdam	2.8/12%	10.5/25%	7.7/6%	5.9/4%	27.3/10%	22.3/3%	9.0/1%	39.3/2%	36.1/0%	23.0/0%	91.5/7%	106.3/0%

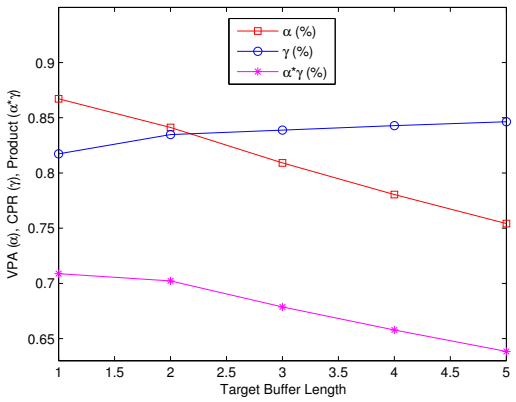

Figure 5: Target Buffer Length Selection. Red curve: average view prediction accuracy (α). Blue curve: average chunk pass rate (γ). Purple curve: product of α and γ. The optimal operation point locates at the peak of purple curve (i.e., $\alpha\gamma$).

can effectively absorb errors in both bandwidth estimation and view prediction, and therefore provides continuous playback with minimum freeze. On the other hand, the received enhancement-tier chunks boost the quality when extra bandwidth is available. Compared with *BS*1, a 3.7x gain in delivered VRR is achieved on average. The delivered VRR margin between proposed TTS and *BS*2 is primarily caused by the base-tier representation. Specifically, our BT is coded to cover the entire 360 video scope with fixed rate R_0 = 10 Mbps and only 1/6 of the total base-tier rate contributes to the delivered VRR, resulting in an initial loss of 8 Mbps. However, when the network average throughput is large, this initial margin becomes negligible and our proposed TTS outperforms *BS*2, as shown from Trace 4 result in Table 2. Besides, with the prefetched base-tier, TTS is much more robust against sudden bandwidth decrease and view prediction error than *BS*2, and therefore has much lower video freeze ratio (VFR).

6 CONCLUSIONS

In this paper, a novel buffer-based two-tier 360 degree video streaming framework is proposed to improve the bandwidth utilization while simultaneously accommodating the user viewing direction changes. Through our trace-driven simulations, a 3.7x gain in video rendering rate is achieved on average compared with traditional 360 video streaming solution. For the future work, we will (1) explore the algorithms to improve the view and bandwidth prediction accuracies, (2) improve two-tier chunk scheduling schemes, (3) extend the current framework to multi-tier framework, and (4) develop quality metric to quantify the delivered video quality.

ACKNOWLEDGMENTS

The authors would like to acknowledge NYU WIRELESS and the affiliate sponsors for their support of this research.

REFERENCES

[1] 2014. OpenTrack: head tracking software. (2014). https://github.com/opentrack/opentrack
[2] 2015. Google Cardboard. (2015). https://vr.google.com/cardboard
[3] M. Budagavi, J. Furton, G. Jin, A. Saxena, J. Wilkinson, and A. Dickerson. 2015. 360 degrees video coding using region adaptive smoothing. In *Proc. IEEE Int. Conf. Image Processing (ICIP)*. IEEE, 750–754.
[4] G. Van der Auwera, Hendry M. Coban, and M. Karczewicz. 2016. *Truncated Square Pyramid Projection (TSP) For 360 Video, JVET Doc. D0071.*
[5] F. Duanmu, E. Kurdoglu, Y. Liu, and Y. Wang. 2017. View Direction and Bandwidth Adaptive 360 Degree Video Streaming using a Two-Tier System. In *Proc. IEEE International Symposium on Circuits and Systems (ISCAS '17)*. IEEE, Baltimore, MD, USA.
[6] Eymen Kurdoglu, Yong Liu, Yao Wang, Yongfang Shi, ChenChen Gu, and Jing Lyu. 2016. Real-time Bandwidth Prediction and Rate Adaptation for Video Calls over Cellular Networks. In *Proceedings of the 7th International Conference on Multimedia Systems (MMSys '16)*. ACM, New York, NY, USA.
[7] Evgeny Kuzyakov. 2015. Under the hood: Building 360 video. (2015). https://code.facebook.com/posts/1638767863078802/under-the-hood-building-360-video/
[8] Evgeny Kuzyakov. 2016. Next-generation video encoding techniques for 360 video and VR. (2016). https://code.facebook.com/posts/1126354007399553/next-generation-video-encoding-techniques-for-360-video-and-vr/
[9] H.-C. Lin, C.-Y. Li, J.-L. Lin, S.-K. Chang, and C.-C. Ju. 2016. *An efficient compact layout for octahedron format, JVET Doc. D0142.*
[10] 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 (ATC '16)*. ACM, New York, NY, USA, 1–6.
[11] Iraj Sodagar. 2011. The mpeg-dash standard for multimedia streaming over the internet. *IEEE Multimedia* 18, 4 (2011), 62–67.
[12] Guibin Tian and Yong Liu. 2012. Towards agile and smooth video adaptation in dynamic HTTP streaming. In *Proceedings of the 8th international conference on Emerging networking experiments and technologies*. ACM, 109–120.
[13] Y.-K. Wang, Hendry, and M. Karczewicz. 2016. *Tile based VR video encoding and decoding schemes, JCTVC Doc. X0077.*
[14] C. Zhang, Y. Lu, J. Li, and Z. Wen. 2016. *segmented sphere projection (SSP) for 360-degree video content, JVET Doc. D0030.*
[15] Minhua Zhou. 2016. *A study on compression efficiency of icosahedral projection, JVET Doc. D0023.*