Vantage: Optimizing video upload for time-shifted viewing of social live streams

Devdeep Ray
Carnegie Mellon University
devdeepr@cs.cmu.edu

Jack Kosaian
Carnegie Mellon University
jkosaian@cs.cmu.edu

K. V. Rashmi
Carnegie Mellon University
rvinayak@cs.cmu.edu

Srinivasan Seshan
Carnegie Mellon University
srini@cs.cmu.edu

ABSTRACT

Social live video streaming (SLVS) applications are becoming increasingly popular with the rise of platforms such as Facebook-Live, YouTube-Live, Twitch and Periscope. A key characteristic that differentiates this new class of applications from traditional live streaming is that these live streams are watched by viewers at *different delays*; while some viewers watch a live stream in real-time, others view the content in a time-shifted manner at different delays. In the presence of variability in the upload bandwidth, which is typical in mobile environments, existing solutions silo viewers into either receiving low latency video at a lower quality or a higher quality video with a significant delay penalty, without accounting for the presence of diverse time-shifted viewers.

In this paper, we present Vantage, a live-streaming upload solution that improves the overall quality of experience for diverse time-shifted viewers by using selective quality-enhancing retransmissions in addition to real-time frames, optimizing the encoding schedules to balance the allocation of the available bandwidth between the two. Our evaluation using real-world mobile network traces shows that Vantage can provide high quality *simultaneously* for both low-latency and delayed viewing.

For delayed viewing, Vantage achieves an average improvement of 19.9% over real-time optimized video streaming techniques across all the network traces and test videos, with observed gains of up to 42.9%. These benefits come at the cost of an average drop in real-time quality of 3.3%, with a maximum drop of 7.1%. This represents a significant performance improvement over current techniques used for SLVS applications, which primarily optimize the video upload for real-time viewing.

CCS CONCEPTS

• **Information systems** → **Multimedia streaming**;

KEYWORDS

Video delivery, adaptive bitrate algorithms, live streaming, VOD

SIGCOMM '19 August 19–23, 2019 Beijing, China

ACM ISBN 978-1-4503-5956-6/19/08…$15.00
https://doi.org/10.1145/3341302.3342064

ACM Reference Format:
Devdeep Ray, Jack Kosaian, K. V. Rashmi, and Srinivasan Seshan. 2019. Vantage: Optimizing video upload for time-shifted viewing of social live streams. In *SIGCOMM '19: 2019 Conference of the ACM Special Interest Group on Data Communication, August 19–23, 2019, Beijing, China.* ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/3341302.3342064

1 INTRODUCTION

Mobile live video traffic has grown significantly over the last decade [14]. This growth has been propelled by improvements in mobile camera technology, computing power, and wireless technology, which enables the capture, encoding, and transmission of high-quality video in real-time from mobile devices. Applications for video-conferencing and live broadcasting have become ubiquitous on mobile devices today. Social live video streaming (SLVS) applications like Facebook Live [1], YouTube Live [11], and Periscope [7], are a new and increasingly popular class of applications that bring the power of live streaming to individuals.

A unique feature that differentiates SLVS platforms from traditional live video applications is the ability to view real-time, time-shifted, and archival versions of a single stream. SLVS applications enable viewers to interact with broadcasters via comments and reactions in real-time [40, 41], and also archive the video to enable viewing after the live streaming session has ended. Furthermore, platforms like Hangouts-on-air [2] and Facebook Live [4] allow multiple users to broadcast simultaneously on a single live-stream. For viewers using interactive features such as real-time comments, as well as broadcasters taking part in collaborative broadcasting [2, 4], it is critical to deliver the video stream at a low-latency, whereas a higher streaming latency is acceptable for the other viewers.

In contrast, traditional live video streaming applications target a *single* viewing delay. In video conferencing applications, participants require low-latency for interactivity, while viewers of a broadcast event can typically tolerate tens of seconds of delay. Existing approaches for handling network bandwidth variations are *tailored for one particular viewing delay*. In low-latency applications like videoconferencing, video bitrate is chosen to closely follow the available bandwidth in order to ensure that frames are received before their real-time playback deadline, at the expense of lower quality during periods of low bandwidth. On the other hand, when higher delays are acceptable, applications use buffers to absorb network variations and the video bitrate is chosen to match the average bandwidth. This results in higher video quality and smoother playback at the cost of higher latency.

Due to lack of better alternatives, current SLVS platforms make the same tradeoffs between latency and quality as traditional live streaming, despite the diversity of viewing delays. Operating at a single point on the latency-quality tradeoff spectrum is inadequate for providing high quality-of-experience (QoE) for all the viewers of SLVS streams. The problem is further exasperated by the fact that SLVS streams are commonly initiated from mobile devices, which have particularly unpredictable network behavior [44].

In this paper, we present Vantage, a live video upload solution explicitly designed to address the time-shifted viewing characteristic of SLVS in the face of bandwidth variations. Vantage exploits the variability of the upload path to it's advantage: periods with high bandwidth can be used to correct for a loss in quality due to previous periods of network impairment. Vantage optimizes the video upload process across different time-shifted viewing delays by using *quality-enhancing retransmissions* in conjunction with a low-latency video stream. Vantage formulates bitrate selection and transport scheduling as a joint optimization problem that maximizes the video quality across the diverse viewing delays.

Several challenges need to be addressed to make the use of quality-enhancing retransmissions practical and effective: (1) allocating bandwidth and scheduling transmissions for the real-time and retransmitted frames such that the QoE is optimized for all users, (2) handling the computational overheads and latencies associated with complex optimization decisions and video compression, and (3) dealing with the network unpredictability and its impact on scheduling decisions. Vantage incorporates several system design choices like approximations, pipelining, and fallback mechanisms to handle the challenges related to optimization and unpredictability.

We have implemented Vantage and evaluated it on a wide variety of mobile network traces [31] and videos [10]. Our evaluation shows that Vantage achieves high quality for low-latency and time-shifted viewing *simultaneously*. Specifically, for delayed viewing, Vantage achieves an average improvement of 19.9% over real-time optimized streaming techniques across all the network traces and test videos, with observed gains of up to 42.9%. The quality achieved by Vantage for delayed viewing is within 7.7% on average of the maximum quality achievable by delay tolerant techniques. These benefits come at the cost of an average drop 3.3% in the real-time quality, with a maximum drop of 7.1%. These results demonstrate the significant performance benefits of using Vantage over current techniques used for SLVS applications, which primarily optimize the video upload for real-time viewing.

2 BACKGROUND AND OPPORTUNITY

In this section, we discuss SLVS architectures and the network variability observed on mobile upload paths used in SLVS.

2.1 SLVS Architectures

We describe common designs and practices employed by four of the most widely used SLVS platforms as of 2019: Facebook Live, YouTube Live, Twitch, and Periscope. Our descriptions are informed by recent studies [40] and industry engineering material [12, 36].

Live video is captured and encoded by a broadcaster's device (e.g., a mobile phone) and uploaded via RTMP [8] or WebRTC [9] to an ingestion point (a point-of-presence or a data center server), where the upload path connection is terminated. We refer to this ingestion point as the *upload endpoint*. The video is then re-encoded at various bitrates and handed off to a content delivery network, which delivers the video to viewers using a variety of techniques that have been widely studied in the past [23, 24, 30]. This paper focuses on improvements for the *upload path* of SLVS applications.

2.2 Time-shifted viewing in SLVS

SLVS differs from traditional live-streaming in that it enables viewing of the same video stream at different delays. Traditional live streaming applications are tailored either for interactive, low-latency streaming (e.g., Skype and Hangouts) or for high quality viewing at larger delays (e.g., ESPN and CNN). On the other hand, SLVS platforms enable both real-time and delayed viewing of the same stream. We term this characteristic "time-shifted viewing."

Time-shifted viewing takes a number of forms within an SLVS stream. Some viewers interact with broadcasters via comments and reactions, and thus require real-time latencies [40, 41]. SLVS platforms also archive video streams to allow for viewing after the live stream has ended, and also allow viewers to seek back to older segments during the live stream and watch the video with a time-shifted delay. Moreover, collaborative broadcasting platforms like Hangouts-on-air [2] and Facebook Live [4] allow collaborative streaming where the co-broadcasters have stronger low latency requirements compared to the viewers.

In summary, SLVS streams have audiences with a wide variety of viewing delays, and thus have varying degrees of latency tolerance. This presents a new and important dimension for improving the quality of experience of SLVS platforms.

2.3 Variability in the upload path

Many SLVS broadcasts are initiated from mobile devices over cellular networks like LTE, which experience frequent bandwidth fluctuations [44]. This forces the broadcaster's device to either adapt the bitrate of encoded video (i.e., alter quality), or use large sender side buffers (i.e., alter the delay of transmission).

To illustrate the variability of bandwidth in mobile uplinks, we analyzed the network traces from the Mahimahi [31] project. Across the eight upload traces, we made the following observations:

(1) **Periods of low/high bandwidth are common:** 17.1% and 17.4% of the time, upload bandwidth is 50% or less and 150% or more than the average for a particular trace, respectively.

(2) **Periods of low/high bandwidth are short-lived:** Periods with less than 50% and more than 150% of the average bandwidth last, on average, 789 ms and 809 ms, respectively.

(3) **More bandwidth is gained during high periods than is lost during low periods:** In five out of the eight traces, we find that there is at least 1.25× additional bandwidth when the bandwidth is above 150% of the average than the amount lost when bandwidth drops below 50% of the average.

These observations suggest that periods of high bandwidth can be exploited to improve the quality of frames that were previously affected by periods of low bandwidth. In the next section, we show how an SLVS upload solution can make use of these properties to improve the QoE for all time-shifted viewers.

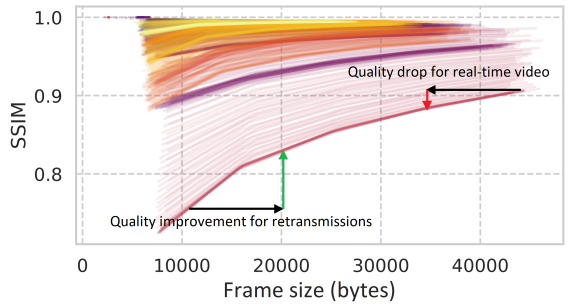


Figure 1: SSIM as a function of the frame size for the Sintel trailer [10, 26]. Each line corresponds to a single frame. The video was encoded multiple times with different target bitrates to generate the data.

3 SUPPORTING TIME-SHIFTED VIEWING

In this section, we show that conventional live-streaming upload techniques are inadequate for providing high QoE for applications that support time-shifted viewing. We then describe an approach for providing high QoE for viewers at different time-shifted delays.

3.1 Inadequacy of existing techniques

We use a simple example to demonstrate the inadequacy of existing live streaming upload techniques at providing high QoE for diverse time-shifted viewers. Consider a broadcaster capturing and uploading an SLVS stream to an upload endpoint. Consider a 20 second period where the upload bandwidth between the broadcaster and the upload endpoint is 0.5 Mbps during the first 10 seconds and 3 Mbps during the final 10 seconds, as depicted in Figure 2a.

The structural similarity (SSIM) index is a common metric used to measure the perceptual quality of video frames. The relationship between video bitrate and SSIM is typically observed to be a concave, non-decreasing function [39]. Figure 1 shows the relationship between the frame size and the SSIM for each frame of the Sintel [10, 26] trailer. To keep the discussion in this section simple, we use a hypothetical model that reflects this relationship between the video bitrate (b) and the SSIM (Q). Specifically, we assume $Q = 1 - \frac{1}{2b+1}$ for each video frame. In Sections 3.1.1 and 3.1.2, we analyze the SSIM of the video received at the upload endpoint when transmitted using conventional live upload techniques.

3.1.1 ***Uploading for delayed viewing.*** We first consider uploading a live stream using techniques commonly used for applications like ESPN and CNN, which typically deliver live content to viewers with up to few tens of seconds of delay. It is common for such techniques to adapt the video bitrate slowly in response to changes in the network bandwidth, and to buffer frames when there is insufficient bandwidth for transmission at the target bitrate. There has been a significant amount of work [22, 25, 32] in the space of bitrate adaptation and buffer allocation for video streaming applications that do not have strict low latency requirements.

For the sake of this example, assume that the uploading client has knowledge of the average bandwidth during the next 20 seconds, which in this case is 1.75 Mbps. The client thus encodes each frame at 1.75 Mbps. Figure 2b (purple dotted line) depicts the video quality using this strategy for delayed viewing. Between 0 s and 10 s, the available bandwidth is lower than the target bitrate. The uploading client consequently buffers frames during this period and begins draining the buffer when the available bandwidth increases at 10 s. This results in an average SSIM of 0.777 for the entire video when viewed at delays of 20 seconds or more.

3.1.2 ***Uploading for real-time viewing.*** We next consider videoconferencing applications like Skype and Hangouts, which use live-streaming techniques that are tailored for real-time viewing. The technique of buffering frames in the face of bandwidth drops described in Section 3.1.1 is inadequate for this setting because buffering delays the transmission of frames beyond the real-time deadline required for interactivity. Real-time video streaming solutions like WebRTC [9] significantly underutilize bandwidth [15] to ensure timely delivery of video frames. Salsify [18] explores video encoding techniques that can accurately match the network bandwidth to reduce buffering for low latency playback.

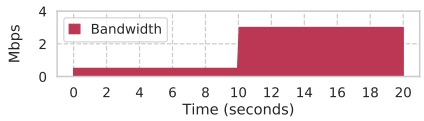
Assuming that the uploading client has accurate instantaneous upload bandwidth estimates, Figure 2b (yellow crossed line) depicts the real-time video quality when using streaming techniques optimized for low latency. Between times 0 s and 10 s, frames are encoded at 0.5 Mbps to minimize buffering delay. When the bandwidth increases at 10 s, frames are encoded at 3 Mbps. This results in an average SSIM of 0.679 across the entire received video.

3.1.3 ***Uploading for more than one viewing delay?*** While the techniques for live video upload described in Sections 3.1.1 and 3.1.2 achieve high QoE for a single viewing delay, they are inadequate for applications where the video is viewed at different delays. Tailoring upload for delayed playback (i.e., Section 3.1.1) results in high QoE for viewing beyond a certain delay, but renders the video unplayable at smaller time-shifts. On the other hand, while real-time streaming techniques (i.e., Section 3.1.2) are suitable for low-latency viewing, this limits the QoE for viewers at larger time-shifts: in the example above, upload optimized for real-time viewers results in an average SSIM of 0.679, whereas uploading for delayed viewing results in an average SSIM of 0.777.

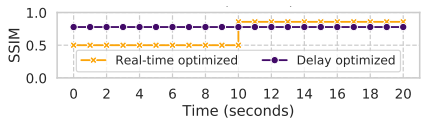
As described in Section 2.2, SLVS applications enable time-shifted viewing in which viewers can watch the same video at different delays. Thus, optimizing live video upload for a single viewing delay is inadequate for SLVS applications, necessitating changes in the architecture of the live video upload process to *improve the quality of experience for both real-time and time-shifted viewers.*

3.1.4 ***Current SLVS platforms do not cater to time-shifted viewing.*** In this section, we demonstrate that common SLVS platforms (Facebook Live, YouTube Live, Twitch, and Periscope) today do not explicitly account for time-shifted viewing, thereby presenting significant opportunities for improving the overall QoE. All the platforms we consider support archival of the live stream for viewing after the termination of the live streaming session. Hence, these platforms support at least two distinct "time-shifted viewing modes": real-time viewing during the live stream, and video-on-demand (VoD) style viewing after the live stream has ended.

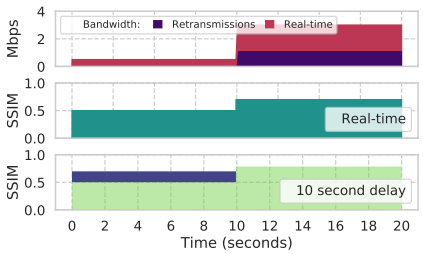
We test whether a platform improves the QoE for time-shifted viewing by introducing a 3-second network impairment on the



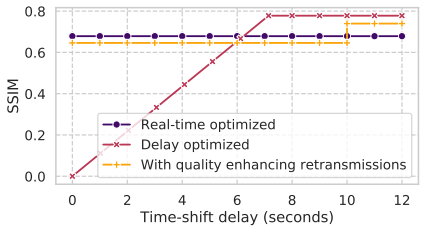
(a) Example bandwidth trace.



(b) SSIM for conventional real-time and delayed streaming techniques.



(c) Bandwidth utilization and improvement in video quality with quality-enhancing retransmissions.



(d) Comparison of average SSIM for different viewing delays for conventional techniques and the proposed technique.

Figure 2: Toy example demonstrating the benefits of quality-enhancing retransmissions over conventional techniques.

upload path. This is followed by 30 seconds of unimpaired transmission. We measure the quality of both the video received by a real-time viewer, and of the version archived by the platform. The full experimental setup is described in Appendix A.

Experimental results. The impairment introduced in this experiment mirrors that described in the toy-example from Section 3.1: there is ample opportunity after the network impairment for improving the video quality for the archived version. Yet, in all experiments, we find that the video frames in the real-time and the archived videos are identical. Both videos experience identical frame drops, resulting in pauses in the received video, and the received frames have identical SSIM values.

These experiments suggest that current SLVS platforms do not exploit opportunities to improve the QoE for more than one viewing delay, leaving considerable opportunity for improving QoE for viewers across diverse viewing delays.

3.2 Proposed approach

We now describe how a live upload solution can provide high QoE for multiple viewing delays.

3.2.1 ***Naïve approach: re-uploading an entire stream.*** A simple strategy to improve the QoE for archived viewing is to store a high quality version of the captured video on the broadcaster's device and re-upload the video after the stream has ended. However, this approach only improves the archival quality; no improvements are achieved for time-shifted viewers during the live stream. Further, this approach consumes a large amount of storage on the broadcaster's device and consumes at least twice as much bandwidth as one would use without re-uploading the stream. These downsides are especially of concern for longer SLVS sessions which may last for multiple hours [27, 41], and for video streams initiated from mobile devices, which often have limited, metered bandwidth and insufficient internal storage.

3.2.2 ***Quality-enhancing retransmissions.*** As described in Section 2.3, mobile networks commonly experience significant bandwidth fluctuations, *both* low and high. Thus, low quality real-time frames transmitted during periods of low bandwidth can be retransmitted *at a higher bitrate* during a later period of high bandwidth *while the live streaming session is ongoing*, thus improving the video quality for delayed, time-shifted viewers. We call these retransmitted frames "quality-enhancing retransmissions" because they are retransmitted for the purpose of improving the quality of a frame compared to the initial version of the frame transmitted for real-time viewing. Unlike typical retransmissions, quality-enhancing retransmissions are not in response to packet loss.

Sending quality-enhancing retransmissions requires reducing the bitrate of real-time frames for allocating sufficient bandwidth for high-quality retransmissions. Due to the concave nature of bitrate-SSIM curves of common video encoding techniques [39], the bitrate of real-time frames can be reduced without causing a significant drop in the video quality. This is evident from the bitrate-SSIM data shown in Figure 1. The drop in quality when reducing the frame size of a high quality frame (solid red arrow) is significantly smaller than the corresponding gain in quality when increasing the frame size of a low quality frame (solid green arrow) by the same amount.

Sending quality-enhancing retransmissions comes at the expense of sending redundant bits over the network and introducing additional computation for frames that have already been sent at a lower quality. The use of scalable video coding (SVC)[35] techniques may result in better performance in some cases. We discuss the implications of using an SVC based design in Section 4.6.

Example of improvements for time-shifted viewing. Consider the example discussed in Section 3.1. Under the approach of using quality-enhancing retransmissions, when the bandwidth is low ($t = 0$ s to $t = 10$ s), the video is encoded at 0.5 Mbps. This behavior is identical to the real-time upload approach. When bandwidth is higher ($t = 10$ s to $t = 20$ s), older frames can be retransmitted at a higher quality as quality-enhancing retransmissions along with the real-time frames. The bitrate of the real-time frames needs to be lower than the available bandwidth (3 Mbps) to accommodate the extra network traffic from the quality-enhancing retransmissions.

The choice of bandwidth allocation between real-time frames and quality-enhancing retransmissions can be driven by high-level goals such as maximizing the viewer-count weighted SSIM across the time-shifted delays. Choosing the optimal tradeoff between the drop in real-time quality and the improvement in delayed video quality can be envisioned as a quality (SSIM) maximization problem across the time-shifted delays. Assuming equal weights for real-time quality and the time-shifted viewing quality at a viewing delay of 10 seconds, the bandwidth allocation which maximizes the average quality across the two delays is 1.84 Mbps for real-time frames and 1.16 Mbps for retransmitted frames. This bandwidth split is depicted in the top plot (bandwidth) in Figure 2c. This results in an average SSIM of 0.643 and 0.742 for the real-time and the delayed viewers, respectively. This is demonstrated in the bottom two plots in Figure

2c. The dark green region in the middle plot (labeled "Real-Time") represents the real-time video quality. The dark blue region in the bottom plot (labeled "10 second delay") shows the improvement in video quality for delayed viewing over the real-time video quality (shown in light green for reference). Figure 2d shows the average quality for time-shifted viewing between $t = 0$ s (i.e., real-time) and $t = 12$ s for video uploaded by real-time optimized, delay-optimized, and time-shift-aware upload strategies. Compared to real-time optimized streaming, using Vantage improves the SSIM for delayed viewing by 9.4%, while causing a drop of only 5.2% in the SSIM of the real-time stream. Note that the drop in SSIM only occurs for the high-quality frames, which still results in good real-time quality as opposed to delay-optimized streaming, where real-time viewing is not feasible. We note here that although we plot the average SSIM in this case to demonstrate the benefits of time-shift optimized streaming, for our evaluation, we use a unified metric (described in Section 5.1.1) that combines the SSIM of the video frames and the stalling events into a single metric.

Thus, by making use of quality-enhancing retransmissions and allocating bandwidth between diverse time-shifted viewing delays, a SLVS upload solution can deliver high QoE to viewers across a spectrum of viewing delays.

3.2.3 ***Key challenges.*** While the idea of sending quality-enhancing retransmissions by exploiting periods of high bandwidth during the live streaming session is simple and promising, there are several critical challenges in designing a live streaming upload solution based on this idea.

Optimal, efficient bandwidth allocation in real time. As described in Section 3.2, allocating bandwidth between real-time frames and quality-enhancing retransmissions can be formulated and solved as an optimization problem. However, finding an optimal solution to this problem may take a non-trivial amount of time, making it challenging to design an optimization-based system involving real-time streaming.

Bitrate-SSIM curve estimation. The bitrate-SSIM curves of frames inform the allocation of bandwidth between real-time frames and quality-enhancing retransmissions. However, the bitrate-SSIM curve of a video frame is not available before the frame is encoded, and thus must be estimated for performing bandwidth allocation. Estimating bitrate-SSIM curves is challenging as they depend on a variety of properties of each frame and the preceding frames.

Mitigating error in bandwidth estimation. Allocating bandwidth between real-time frames and quality-enhancing retransmissions requires having an estimate of the future bandwidth. Accurately estimating future bandwidth is challenging in its own right, and thus a system allocating bandwidth between real-time frames and quality-enhancing retransmissions must be robust in it's abilities to adapt to inaccuracies in bandwidth estimation.

In sections 5.2.7 and 5.2.6, we show that addressing these challenges are critical for improving the QoE across all viewers for time-shifted viewing of live video streams.

4 DESIGN OF VANTAGE

In this section, we describe the design of Vantage and how Vantage overcomes the challenges outlined in Section 3.2.3 in order to deliver high QoE to diverse time-shifted viewers.

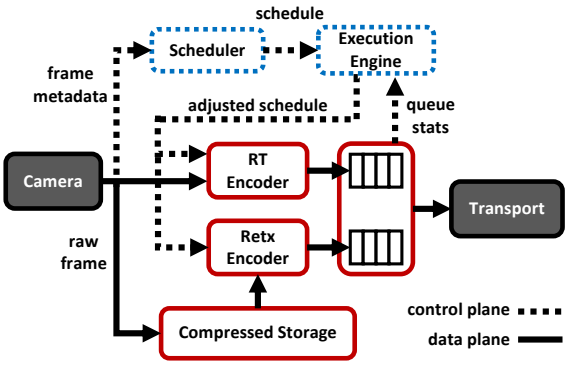


Figure 3: Vantage's architecture. Solid lines indicate data-plane components. Dotted lines indicate control-plane components. Components that are unchanged by Vantage are shown in a darker shade.

4.1 Overview

We first describe the architectures of current live video upload systems, and then describe Vantage's high-level operation.

4.1.1 ***Current live video upload systems.*** In current live video upload systems, the uploading client first captures raw video frames from the camera on a mobile device. Frames are then compressed by an encoder and transmitted to the upload endpoint. The system's network transport mechanism estimates the available upload bandwidth, which informs the level of compression used for encoding the video frames.

4.1.2 ***Architecture of Vantage.*** Figure 3 depicts Vantage's high-level architecture. Vantage modifies the upload architecture described in Section 4.1.1 to enable support for quality-enhancing retransmissions. Raw frames from the camera are encoded and enqueued for real-time transmission. Vantage simultaneously compresses these frames at a high quality and places them in memory for potential quality-enhancing retransmissions in the future. Vantage's scheduler generates a bandwidth allocation schedule for the new frames captured by the camera as well as for quality-enhancing retransmissions. The execution engine coordinates the encoding of scheduled frames, and adjusts for inaccuracies in the allocation determined by the scheduler (discussed in detail below in Section 4.3). Frames that have been scheduled for transmission are enqueued for transmission by a generic transport protocol that is unmodified by Vantage. We assume that the transport layer provides network bandwidth estimates and drains packets from Vantage's queues.

In the remainder of this section, we describe each of Vantage's components in detail as well as how Vantage overcomes the challenges presented in Section 3.2.3.

4.2 Scheduler design

Vantage's scheduler takes as input a set of real-time frames and potential candidate frames for retransmissions, and an estimate of the upload bandwidth in the near future in order to choose (a) which frames to schedule for transmission and (b) the bitrate each scheduled frame should be encoded at.

We formulate schedule generation as a *mixed-integer optimization problem* that generates a transmission schedule that optimizes the quality for the viewers across time-shifted delays. The precise formulation of the optimization problem is described in Section 4.2.1.

An important consideration with this approach is that the time taken to solve a mixed-integer problem may be non-trivial and highly variable. To address this, Vantage's scheduler is run every P seconds and generates schedules for the next P seconds. When the scheduler is run at time $T = t$, it receives a snapshot of the state (i.e., candidate frames and bandwidth estimation) at time t, and generates a schedule for the period between $T = t + P$ and $T = t + 2P$. If the optimization takes longer than P seconds, the scheduler is interrupted and the current, potentially sub-optimal solution of the optimization is used as the schedule.

4.2.1 ***Optimization formulation.*** Next, we discuss the formulation of the optimization problem that is performed by the scheduler every P seconds. Consider a single optimization iteration that starts at time $T = t$. We first list the inputs to the optimizer and then subsequently discuss how these inputs are obtained. The optimizer takes the following information as inputs:

- An estimate of the number of bytes B that can be transmitted between $T = t + P$ and $T = t + 2P$.
- A set of future real-time frame ids (F) that will be sent between $T = t + P$ and $T = t + 2P$.
- A set of past frame IDs (G) that are to be considered for retransmission. A retransmission chosen in this iteration would happen at some time between $T = t + P$ and $T = t + 2P$. The difference between $T = t + 2P$ and the time at which a frame g was captured is the delay of the frame, which we denote as d_g.
- The quality of past frames that have been received by the upload endpoint. For each frame $g \in G$, R_g denotes the SSIM of the version of frame g currently available at the upload endpoint. We do not schedule queued or in-flight frames for retransmission.
- The bitrate-SSIM curve for each frame $g \in G$. For each frame $g \in G$, $\mathrm{Q}_g : size \rightarrow ssim$ represents the mapping from encoded frame size (in bytes) to the SSIM.
- The predicted bitrate-SSIM curves for each of the real-time frames that will be sent between $T = t + P$ and $T = t + 2P$. We denote this by $\mathrm{Q}_f : size \rightarrow ssim$.
- The distribution of the viewing delays of the current set of viewers. For each viewing delay d, $\mathrm{N}(d)$ represents the count of viewers watching the live stream at a viewing delay of d seconds.
- We also define a set of weights $w_g \ \forall g \in G$ and a weight w_0 for the real-time frames. We discuss how these weights are computed from the delay distribution N in the subsequent paragraphs.

The scheduler returns the target sizes $s_f \ \forall f \in F$ for the real-time frames and a set of frames $G' \subset G$ and the corresponding target size $s_g \ \forall g \in G'$ for the quality-enhancing retransmissions. We formulate the optimization problem as a maximization of the weighted viewing quality subject to bandwidth constraints.

The role of the bandwidth constraint is to ensure that the total amount of data scheduled for transmission (including both the real-time frames and past frames) does not exceed the estimated bandwidth. Thus, the bandwidth constraint is :

$$\sum_{\forall f \in F} s_f + \sum_{\forall g \in G} s_g \leq B \tag{1}$$

The objective function includes contributions from the real-time frames and the past frames. For a real-time frame $f \in F$, the contribution to the objective function is

$$w_0 \cdot \mathrm{Q}_f(s_f) \tag{2}$$

For a past frame $g \in G$, the contribution to the objective function is

$$w_g \cdot \max(\mathrm{Q}_g(s_g), R_g) \tag{3}$$

Note that the weights w_g are different for each frame $g \in G$.

The net objective sums up the contribution from each of the real-time frames and the past frames:

$$obj = \sum_{\forall f \in F} w_0 \cdot \mathrm{Q}_f(s_f) + \sum_{\forall g \in G} w_g \cdot \max(\mathrm{Q}_g(s_g), R_g) \tag{4}$$

Since the real-time frames serve as a base for delayed playback as well, the transmission of a real-time frame benefits all delays. Similarly, a quality-enhancing retransmissions at a delay d is useful for all viewing delays that are greater than d. Thus, we set the weights for the real-time frames (w_0) and the past frames (w_g) in the objective function as follows.

$$w_0 = \sum_{d} \mathrm{N}(d), w_g = \sum_{d > d_g} \mathrm{N}(d) \tag{5}$$

The functions Q_i that maps size to SSIM for a frame i is typically a non-linear curve (e.g., Figure 1) and can vary significantly across frames. We approximate these curves as piece-wise linear functions in the formulation of the mixed-integer program. Since the number of frames that can be encoded in P seconds is limited, we additionally limit the number of retransmissions to $|F|$. This ensures that the computational requirements of encoding the quality-enhancing retransmissions does not exceed that of the real-time frames.

4.2.2 ***Bitrate-SSIM curve estimation.*** Recall from Section 4.2.1 that the optimization problem uses bitrate-SSIM curves of all frames that are candidates for scheduling (i.e., Q_g for retransmissions and Q_f for real-time frames). This is required so that the optimization can make informed choices when reducing the real-time quality to improve the quality of past frames.

Vantage uses a regression heuristic to estimate these curves from previous encoding data. We use a function of the form $\mathrm{Q}(s) = 1 - \frac{1}{a \cdot s + b}$ since it captures the concave non-decreasing behavior (for $a > 0, s > \frac{-b}{a}$) typical of bitrate-SSIM curves (e.g., Figure 1). Parameters a and b are computed separately for each frame based on its observed size and SSIM when the frame has already been previously encoded (i.e., for real-time or for quality-enhancing retransmissions). These parameters are updated each time a frame is re-encoded for retransmission. However, bitrate-SSIM information is not available for future real-time frames because they have not yet been captured. Hence, for the future frames, we use the EWMA values of the past parameters for computing the parameters of $\mathrm{Q}_f \ \forall f \in F$ because frames that are temporally local have similar content, and thus similar bitrate-SSIM curves.

4.2.3 ***Optimizer performance.*** A preliminary evaluation of Vantage with a scheduling period $P = 2$ s indicated that the mixed-integer solver often fails in finding an optimal solution within 2 seconds when $|G|$ is large. One alternative is to increase the scheduler period P, but this results in worse performance due to the scheduler receiving stale bandwidth estimates. This is further discussed in Section 5.2.5. Instead, Vantage filters G using a heuristic and only generates the variables in the mixed-integer program for the 200 frames with the worst SSIM. Furthermore, we do not restrict the frame sizes to be integers, and instead use an integer approximation for the continuous solution. We find that using $P = 2$ seconds along with these approximations leads to the optimizer generating high-quality schedules: 98.5% of optimization windows in our evaluation result in a schedule that is within 1% of the optimal solution.

4.3 Mitigating bandwidth estimation error

As described in Section 4.2, Vantage's scheduler generates an encoding and transmission schedule for P seconds in the future based on an estimate of the future network bandwidth.

Since the optimizer uses a bandwidth estimate measured P seconds before the scheduled frames are transmitted, the true available bandwidth may differ at the time when the scheduled frames will be transmitted. Left uncorrected, the use of a schedule generated from a mispredicted bandwidth estimate will lead to sub-optimal use of the network.

To mitigate the effects of bandwidth misestimation, Vantage's execution engine makes adjustments to the generated schedule prior to transmitting the frames. When the bandwidth estimate used to generate the schedule under-estimated the amount of bandwidth available at transmission time, Vantage's execution engine keeps the network saturated by increasing the bitrate of the real-time video compared to the optimizer's schedule, but only if the retransmissions scheduled in that iteration have been completed.

On the other hand, when the bandwidth estimate used to generate the schedule over-estimated the amount of bandwidth available at transmission time, the execution engine prioritizes transmission of real-time frames: frames scheduled for retransmission at that time are discarded and real-time frames are encoded at a bitrate lower than that specified by the scheduler so as to avoid over-saturating the network. Prioritizing real-time transmission in the event of bandwidth over-estimation ensures high QoE for all time-shifted viewing delays, as real-time frames would be available for viewing at all delays, whereas retransmitted frames only benefit viewers watching with a time-shifted delay.

4.4 Encoding retransmissions

Frames that have been scheduled for retransmission at a particular time may not be temporally close to the real-time frames scheduled at the same point in time. This presents a challenge for encoding Vantage's output stream because video encoding algorithms rely heavily on the similarity between successive frames to achieve good compression ratios. Using the same encoder for transmitting both the real-time video and the retransmissions would result in poor compression, as the content in the retransmitted frames may differ significantly from real-time frames.

To address this challenge, Vantage uses two separate encoders for compressing real-time and retransmitted frames. Real-time frames are encoded in the order in which they were captured. Quality-enhancing retransmissions are encoded by a separate encoder based on the schedule determined by the optimizer. Though retransmissions could be temporally far from one another, we note that network impairment events commonly affect groups of neighboring frames. Thus, if a particular frame is a good candidate for retransmission, it is more likely that its neighboring frames are also good candidates for retransmission. We, therefore, add an additional regularization objective to the optimization formulation to favor scheduling consecutive sequences of frames among the retransmission candidates for quality-enhancing retransmissions.

4.5 Reducing memory overhead

Vantage keeps previously transmitted frames in memory so that they can be re-encoded as quality-enhancing retransmissions at a later time. Naïvely storing raw video frames in memory is impractical for uploads initiated from a mobile device; the size of a raw frame can be as large as 1.5 MB, thus requiring more than a gigabyte of memory for 30 seconds of video.

To address the high cost of storing raw video frames, Vantage compresses raw frames as lossless I-frames using a tertiary encoder prior to storing in memory. This allows Vantage to maintain a low memory footprint, but incurs additional computational cost. We believe that this is an appropriate trade-off as hardware accelerated encoding and decoding solutions are commonly available today, though we note that this design choice is not required by Vantage's framework.

4.6 Discussion

In this section, we briefly discuss required changes to the upload endpoint to support Vantage and how Vantage would differ with the availability of an SVC codec.

Upload endpoint modifications. Recall from Section 2.1 that an SLVS upload stream is terminated at an upload endpoint, which decodes the stream and re-encodes it into small video segments for efficient delivery to the viewers over content delivery networks (CDNs) [40].

While the changes proposed in Vantage allow backward compatibility with real-time streaming systems, the upload endpoint must be able to handle Vantage's quality-enhancing retransmissions. This requires the upload endpoint to re-transcode past video segments whenever quality-enhancing retransmissions for that segment are received, and to disseminate these higher quality video segments to the CDN. As described in Section 4.4, Vantage's scheduler penalizes retransmissions that are spread apart, which helps limit the rate at which past video segments need to be updated. Requiring only these minor changes to the upload endpoint makes Vantage well-suited for current SLVS architectures.

Scalable video coding. Vantage's approach of sending quality-enhancing retransmissions bears similarity to the enhancement layers used in scalable video coding (SVC) techniques. Indeed, Vantage's design could be simplified by using an SVC codec, since the video would only need to be encoded once, and storage of

high-quality frames would not be necessary. Despite these benefits, we have chosen not to design Vantage specifically for SVC codecs because (1) SVC codecs are not widely adopted, limiting hardware-accelerated encoding support, and (2) while simple SVC schemes with coarse grained scalability do not have significant overhead, fine grained SVC schemes have poor compression efficiency and are significantly more compute intensive compared to non-layered codecs.

Even in the absence of the aforementioned downsides of SVC, the use of SVC alone cannot overcome the challenges involved with optimizing the video upload for multiple time-shifted delays. An SVC based live video upload mechanism would still need to make bandwidth allocation decisions between the real-time base video stream and the enhancement streams, and also choose which frames to retransmit. While the ability to encode the video only once and not having to store high quality frames is an advantage of using SVC, the use of non-layered codecs is better in some situations. When retransmitting a sequence of contiguous frames, encoding a P-frame with a high quality past frame as the reference is often more efficient than encoding the frame with a low quality version of itself as the reference frame.

While SVC provides some clear benefits, we believe compatibility with standardized codecs and hardware is more important for adoption in the real world today. We would only need to make minor tweaks to the optimization formulation used in Vantage's scheduler for generating optimized schedules for SVC codecs.

Overhead of two encoders. Vantage's approach of using two separate encoders to compress real-time frames and quality-enhancing retransmissions is computationally expensive. We believe this overhead is well-justified: Vantage gains significant improvements in the QoE across multiple viewing delays for SLVS applications, and the trend of increasing hardware acceleration support further justifies this tradeoff.

4.7 Implementation details

We have implemented Vantage in C++, with the scheduler using the Gurobi [19] library to solve the optimization problem. To reduce computational requirements, we limit Gurobi to run on a single core and the execution engine to run on a single thread. Vantage uses the VP8 encoder from Salsify [18] because it provides a convenient API for controlling the size of each frame. For performance reasons, Vantage compresses the high-quality frames and encodes real-time frames and the quality-enhancing retransmissions in parallel.

5 EVALUATION

We evaluate Vantage and compare its performance to conventional live video upload techniques for SLVS applications.

5.1 Methodology

5.1.1 ***Metrics.*** Vantage is designed to improve the quality of video playback across the various time-shifted viewing delays by replacing low quality frames with high-quality versions and filling in the gaps caused due to skipped frames. While Vantage's scheduler is designed to optimize the SSIM [42] metric, Vantage can support other frame level reference metrics (like PSNR, etc.). We use the SSIM metric when measuring the quality of a single frame and use the SQI-SSIM [16] metric to compute an overall video quality score from the SSIM values of the individual frames. Most objective video quality metrics do not consider the effect of stalling when calculating video quality [43]. SQI-SSIM is a unified metric that takes into account the full reference quality of each frame and also the duration and frequency of video stalls. SQI-SSIM uses an exponential decay function instead of zeros to fill in the SSIM of missing frames. Thus, shorter stalls have a smaller effect on the overall video quality. When video playback resumes after a stall, the SSIM of the subsequent frames are penalized according to an exponential decay function, thus accounting for the frequency of stalls. We note that Vantage can support other video quality assessment metrics (such as PSNR).

5.1.2 ***Baselines.*** While there is significant prior work on optimizing video streaming for the individual cases of live streaming and VOD-style video streaming, we are not aware of any prior research on optimizing video quality simultaneously for real-time streaming and time-shifted viewing of the streams. Vantage is designed to work with existing congestion control and video coding systems and enhance the performance of these systems for scenarios involving time-shifted viewing of live streams, and is not meant to be a standalone end-to-end solution for SLVS video upload.

We compare Vantage to the best case performance for low latency streaming and VOD-style streaming using an idealized model for bandwidth estimation and congestion control:

Low latency streaming (Base-RT). Existing low latency optimized streaming systems like WebRTC and Skype maintain low latency by conservatively utilizing the network bandwidth to prevent network saturation. Recent approaches like Salsify [18] utilize the network better by matching the instantaneous network estimate through tight coupling of the video encoder and the transport protocol. Base-RT models these systems by encoding individual video frames at a bitrate that closely follows the real-time network estimate. This results in optimal video quality performance for low latency streaming.

Buffered streaming (Base-Delay). The use of larger buffers at the sender enables a streaming application to encode video at the average network bandwidth. This is similar to conventional ABR based video streaming solutions like HLS [3] and MPEG-DASH [5], which split the video into small segments where each segment is encoded at a specific bitrate. To model the characteristics of streaming techniques that use buffers and slower rate adaptation, we use a window of 30 seconds to compute the average bandwidth and encode the video at this bitrate. This results in optimal video quality for cases where a delay of 30 seconds is acceptable.

5.1.3 ***Evaluation setup.*** We evaluate Vantage with a combination of videos and network traces with different characteristics. Our experiments were run on a machine with Intel(R) Xeon(R) processors, limiting the Gurobi [19] solver to a single core for emulating computational limits in mobile environments. Unless otherwise specified, we evaluate Vantage for a uniform time-shifted viewing delay distribution for the optimization described in Section 4.2.1 and use a 2 second period for the optimizer. This choice is further discussed in Section 5.2.5. We evaluate the effects of different distributions of the time-shifted viewing delays in Section 5.2.3. We run the live streams for 150 seconds and ignore the data for the

		Talking Heads			City Panning			Animation		
Trace	Delay	Base-RT	Base-Delay	Vantage	Base-RT	Base-Delay	Vantage	Base-RT	Base-Delay	Vantage
Verizon LTE	Real-time	0.8885	0.5854	0.8750	0.8003	0.6479	0.7792	0.9279	0.7504	0.9225
	30s delay	0.8896	0.9552	0.9438	0.8012	0.8472	0.8306	0.9290	0.9818	0.9834
		(0.8890)	(0.7703)	**(0.9094)**	(0.8008)	(0.7475)	**(0.8049)**	(0.9284)	(0.8661)	**(0.9529)**
AT&T LTE	Real-time	0.5638	0.2236	0.5538	0.5224	0.4008	0.4880	0.6511	0.4370	0.6648
	30s delay	0.5705	0.9098	0.8155	0.5285	0.7198	0.6760	0.6576	0.9600	0.9327
		(0.5672)	(0.5667)	**(0.6846)**	(0.5254)	(0.5603)	**(0.5820)**	(0.6543)	(0.6985)	**(0.7987)**
TMobile UMTS	Real-time	0.4957	0.1942	0.4604	0.3371	0.0965	0.3199	0.5055	0.1390	0.4833
	30s delay	0.5054	0.6774	0.5840	0.3451	0.4834	0.4011	0.5169	0.7322	0.6143
		(0.5005)	(0.4358)	**(0.5222)**	(0.3411)	(0.2899)	**(0.3605)**	(0.5112)	(0.4356)	**(0.5488)**

Table 1: SQI-SSIM achieved by the baselines and Vantage for each combination of the videos and the network traces. In each case, the average SQI-SSIM across delays (indicated within parentheses) is the highest for Vantage (bolded).

last 30 seconds, since the measurements for the last 30 seconds may be affected by the early termination of the program. We repeat videos and traces that are shorter than 150 seconds until the entire simulation is complete.

Videos. We chose three diverse and representative videos from the Xiph.org Test Media repository [10] for our evaluation. Appendix B contains screenshots of each of these three videos. "Talking Heads" contains four people talking in front of a static background. This style of video is the most common among SLVS streams [37] and is typically easier to encode. "City Panning" pans across the city of Stockholm. This video is much harder to encode due to a higher amount of moving content and very fine details. "Animation" is an animated video sequence with varying degrees of motion over the duration of the video, which makes some segments easy to encode, while other parts are harder to encode.

Bandwidth traces. We chose a diverse set of network traces from the Mahimahi [31] repository: a high bandwidth LTE trace, a highly variable LTE trace, and a low bandwidth UMTS trace. We find these traces to be representative of Vantage's performance across all traces in the repository.

Transport layer emulation. We use a bandwidth averaging window of 100 ms for Base-RT and the real-time stream in Vantage. We use the average bandwidth in the past 1 second as the bandwidth estimate to Vantage's scheduler. For Base-Delay, we use the average bandwidth of the previous 30 seconds. We run Vantage and the receiver on the same machine and emulate packet transmissions according to the provided bandwidth trace.

Many live-streaming systems use FEC [21] or packet-level retransmission for dealing with network losses. These techniques can be incorporated into the network model by reducing the bandwidth estimates provided to Vantage and using the excess bandwidth for loss recovery mechanisms (e.g., FEC). Since we evaluate baselines using the same model, this provides a fair comparison between Vantage and existing techniques for live video upload.

Encoder performance. Salsify's encoder is a software-based VP8 encoder written in C++. Software encoders are much slower than hardware based encoders. We observed that even with parallel encoding of the frames, the encoder was not able to achieve a rate of 30 FPS while encoding 1280 × 720 (HD) videos. Hence, we run Vantage with time dilation to allow the encoder to run at 30 FPS in virtual time, but limit the optimization to P seconds of *wall clock time*. This allows us to evaluate Vantage in a manner that is agnostic to the encoding speed of the specific encoder we chose.

Ethics. This work does not raise any ethical issues. We use test video sequences [10] and anonymized bandwidth traces [31] that are publicly available.

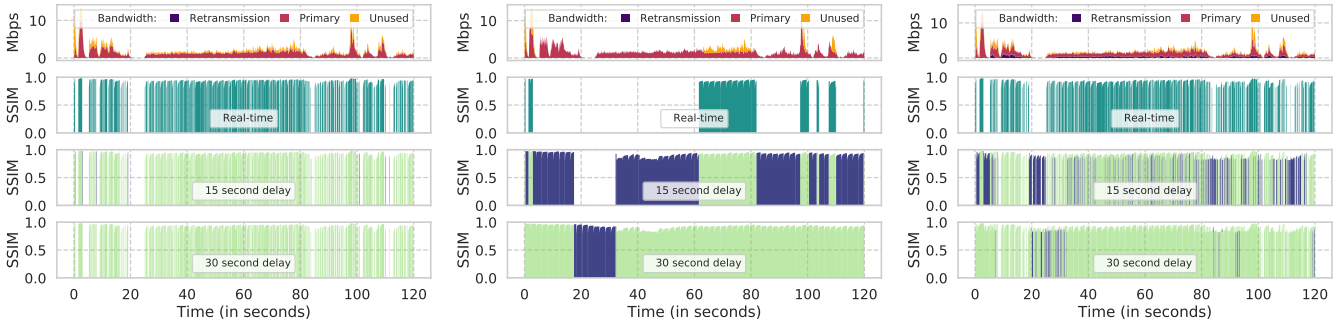
5.2 Results

The highlights of our evaluation are as follows:

- Across a variety of upload bandwidth traces and videos, Vantage improves the SQI-SSIM for time-shifted viewing over Base-RT by 19.9% on average (Section 5.2.1).
- Vantage *simultaneously* achieves high real-time quality (within 3.3% of the quality achieved by real-time optimized streaming techniques on average) and high quality for delayed viewing (within 7.7% of the optimal quality achievable for delayed viewing on average), demonstrating the effectiveness of using Vantage for applications that involve time-shifted viewing such as SLVS applications (Section 5.2.1).
- Vantage is able to adapt and achieve high QoE for skews in the distribution of viewing delays (Section 5.2.3).
- Vantage can also improve the QoE across different viewing delays for videos with highly dynamic (i.e., harder to encode) content, even when there are no bandwidth variations (Section 5.2.4).
- Vantage is robust to bandwidth misestimation (Section 5.2.6).

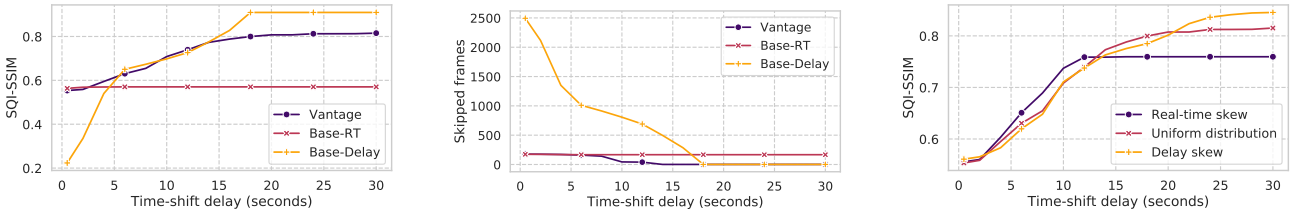
5.2.1 ***Overall improvements.*** Table 1 contains the SQI-SSIM of the received video at both real-time and a delay of 30 seconds for all the traces and videos. The average SQI-SSIM over the two delays (shown in brackets) captures how a particular upload technique caters to both viewing delays.

We observe that in all cases, Vantage achieves significantly better quality than Base-RT (up to 42.9%) for delayed viewing. The gains are biggest for the AT&T-LTE network trace; this can be attributed to the significant variations in the bandwidth. Even for the TMobile-UMTS network trace, which has very low bandwidth and significant periods of zero bandwidth, we observe modest improvements in the delayed viewing quality over Base-RT. This demonstrates that Vantage can *simultaneously* deliver high QoE for both real-time and time-shifted viewing. We also find that, across all traces and videos, Vantage (bolded entries) significantly outperforms both Base-RT and Base-Delay in average SQI-SSIM across the viewing delays.



(a) Base-RT (b) Base-Delay (c) Vantage

Figure 4: SSIM of frames at various viewing delays for the Talking Heads video and the AT&T-LTE trace. The top row shows the utilization of the upload bandwidth. The bottom three rows show the quality of the received video at real-time, 15 seconds, and 30 seconds of delay. The dark green region represents the instantaneous quality for the real-time timeseries. For the 15-second delay and the 30-second delay timeseries, the dark blue region represents the improvement in quality compared to the real-time quality and the 15-second delay quality respectively. The light green shaded regions in the 15-second and 30-second timeseries represent the baseline video quality for the previous delay bucket (real-time and 15-second respectively).



(a) SQI-SSIM achieved by Vantage and the two baselines for various time-shifted delays.

(b) Frames skipped in groups of 10 or more as a function of the viewing delay.

(c) Performance of Vantage for various viewing delay distributions.

Figure 5: Fine-grained time-shift results for the Talking Heads video and AT&T-LTE trace.

As discussed in Section 3.2, the cost paid by Vantage is slightly worse quality for real-time-only and delayed-only viewing as compared to upload transmission techniques optimized individually for either of these settings. Across all traces and videos, we find that the SQI-SSIM of real-time video resulting from Vantage is no more than 7.1% worse than that resulting from Base-RT, with the improvement in the SQI-SSIM for delayed viewing being higher than the drop in the real-time SQI-SSIM in all cases.

We note that while Base-RT and Base-Delay are specialized for real-time and delayed viewing, they are *not necessarily* optimal for these targets. Vantage can occasionally outperform the baselines for these targets (as is seen in Table 1) for the following reasons: (1) Base-RT occasionally sends real-time frames that are larger than the available bandwidth, which causes frames to be dropped. In contrast, Vantage encodes real-time frames more conservatively because it also performs retransmissions. (2) Base-Delay can build large buffers of frames that take longer than 30 seconds to drain, while Vantage only schedules retransmissions that are expected to go through in P seconds.

5.2.2 ***Inspecting Vantage's improvements.*** In this section, we analyze Vantage's improvements over Base-RT and Base-Delay in detail for a single video and bandwidth trace combination (AT&T-LTE trace and Talking Heads video). Figure 4 compares the baseline approaches to Vantage by plotting the raw SSIM of each frame of the video for various viewing delays. Frames not received in time for a particular viewing delay are shown as zero.

Base-RT (Figure 4a) achieves reasonable quality for real-time viewing, skipping frames when the network bandwidth is zero, but does not improve quality for higher viewing delays. Whether a viewer views the stream in real-time or at a delay of 30 seconds, they would both experience a 5-second pause in the video starting at approximately $t \approx 20$ s.

Base-Delay (Figure 4b) on the other hand achieves smooth, high-quality video playback for viewing delays of more than 30 seconds, but is unplayable at smaller time-shifts since a majority of the video does not get delivered in time for real-time playback. Even at a delay of 15 seconds, there is a large pause in the video playback between times $t \approx 18$ s and $t \approx 33$ s.

Vantage (Figure 4c) has no noticeable drops in the real-time video quality as compared to Base-RT. The added benefit of using Vantage can be seen for higher viewing delays where the video quality is dramatically improved. One example is the 5 s period starting at $t \approx 20$ s, where the network bandwidth is zero. We observe in Figure 4c that for viewing delay of 30 s and higher, Vantage repairs this entire segment using quality-enhancing retransmissions, thus resulting in significantly improved video quality over Base-RT. Vantage achieves this by reducing the amount of bandwidth used

for real-time frames slightly, and using the excess bandwidth for sending the high-quality retransmissions, as shown in the bandwidth usage graph in the top subplot in Figure 4c. With Vantage, the delayed viewing quality comes very close to that of Base-Delay, while *simultaneously* achieving real-time playback quality that is comparable to Base-RT.

Improvements at fine-grained time shifts. Figure 5a plots the SQI-SSIM of the video for each viewing delay. For real-time viewing, Base-RT outperforms both Base-Delay and Vantage. At viewing delays beyond 15 seconds, Base-Delay outperforms both Base-RT and Vantage. While the performance of Base-RT and Base-Delay is unsatisfactory for the viewing delays that they are not optimized for, Vantage provides a smooth increase in quality as time-shift delay increases. Vantage's performance is competitive simultaneously at the delays for whith the two baselines are separately optimized for.

Figure 5b shows the number of frames of the received video that are skipped in groups of 10 or more frames for different delays. This quantifies the smoothness of the resulting video. Both Base-RT and Base-Delay suffer from a large number of skipped frames for time-shifts that they are not optimized for. In contrast, Vantage has nearly the same number of skipped frames as the baselines at their respective optimal delays and significantly reduces the number of skipped frames for intermediate delays.

5.2.3 ***Adapting to viewer-delay distributions.*** An important aspect of Vantage is the ability to optimize the video upload process for different distributions of the viewing delays. We evaluated the performance of Vantage for three different viewing delay distributions, one skewed towards real-time viewing, one skewed towards delayed viewing and one with uniform weights for low latency and delayed viewing. Figure 5c shows the effect of these weights on the quality of video at different viewing delays between 0 and 30 seconds. For low viewing delays between 0 and 10 seconds, Vantage with a real-time skewed delay distribution achieves the highest quality, whereas for highr viewing delays between 22 and 30 seconds, Vantage with a delay skewed distribution achieves the highest quality. Vantage with a uniform delay distribution strikes a balance between the two across all delays, achieving the highest quality for viewing delays between 10 and 22 seconds. This demonstrates that Vantage can not only support multiple time-shifted viewing delays, but also be tuned to cater to the exact distribution of the viewing delays for optimized QoE across the different delays.

5.2.4 ***Quality improvements for dynamic videos.*** In addition to improving QoE in the face of bandwidth variations, Vantage can also be used to compensate for lower video quality for harder to encode segments of a video, even when there is no bandwidth variation. Video content with varying compression difficulty is common in video game streaming applications like Twitch, where the video is significantly harder to encode during highly dynamic segments compared to more static segments like in-game menus. To emulate this setting, we run Vantage and Base-RT for the Animation video, which is an animated sequence with highly dynamic scene content, with a constant bandwidth of 1.5 Mbps.

Base-RT causes 7 frames to be dropped. These drops not only affect real-time viewing, but are also present during delayed viewing. On the other hand, Vantage drops 8 frames in real-time, but retransmits these later *during the live-stream*, resulting in *no lost frames for delayed playback* and a corresponding increase in the SQI-SSIM from 0.960 (for Base-RT) to 0.963.

5.2.5 ***Optimizer period.*** We ran Vantage with different values of P ranging from 1 second to 8 seconds, and the results are shown in Figure 6a. Choosing a large time period allows the optimizer to spread the retransmissions over a longer duration, resulting in better real-time quality, but smaller improvements for delayed viewing since the bandwidth estimates are stale. Smaller time periods result in more accurate bandwidth estimates, but choosing a time period that is too small results in a bigger drop in the real-time quality and smaller improvements for delayed viewing due to retransmissions being squeezed into shorter periods.

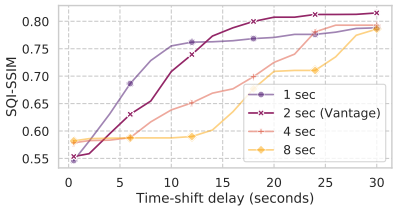
5.2.6 ***Errors in bandwidth estimation.*** Recall from Section 4.3 that Vantage's scheduler uses the average bandwidth from the previous 1 second to schedule frame transmissions for a time period that is 2 seconds in the future. To evaluate the effect of bandwidth misestimation, we analyze Vantage's performance when given consistently erroneous bandwidth estimates on the AT&T-LTE trace with the Talking Heads video.

Figure 6b depicts the SQI-SSIM for different time-shifted delays when Vantage is faced with varying degrees of bandwidth estimation error. Vantage in its normal operation mode (i.e., using the average bandwidth of the past) is listed as "Past estimate (Vantage)." We also emulate Vantage for the hypothetical scenario where it knows the exact future bandwidth, along with a 50% underestimate and a 100% overestimate from the future bandwidth (labeled accordingly in Figure 6b).

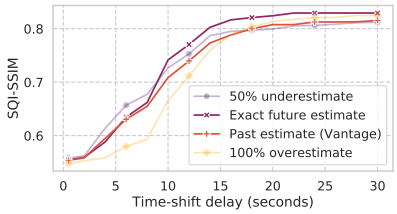
We see that Vantage achieves only slightly lower SQI-SSIM values across times-shift delays compared to what it would achieve with knowledge of the exact future bandwidth. Across all video-trace combinations, we find that Vantage results in drops in quality of at most 1.7%, 3.2%, and 2.4% for real-time, 15 second, and 30 second viewing delays, as compared to what it would achieve with knowledge of the exact future bandwidth. This suggests that Vantage's approach of using the average bandwidth from the previous 1 s is satisfactory for generating high-quality bandwidth allocation schedules and Vantage is resilient to errors in bandwidth estimation.

Figure 6b further indicates that Vantage is highly resilient to even large bandwidth estimation errors due to effective fallback mechanisms: Vantage achieves high SQI-SSIM even when the bandwidth is severely misestimated. The real-time video is largely unaffected by bandwidth misestimation in the scheduler since the execution engine adapts to rapid variations in the real-time bandwidth (described in Section 4.3). When the generated retransmissions are too large to be transmitted, the scheduler overwrites the schedule in the next iteration. On the other hand, when the generated retransmissions are small, they get transmitted faster and the excess bandwidth is allocated to real-time frames. We observe similar patterns for the other two network traces. Additional graphs and discussion are included in Appendix C.

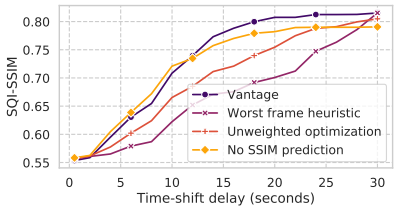
5.2.7 ***Ablation studies.*** The use of a quality enhancing retransmissions to improve the video quality for higher viewing delays can be implemented in multiple ways and it is important to understand the additional benefits Vantage's design provides over



(a) Effects of choosing different optimizer periods on the video quality. Vantage uses $P = 2$ sec



(b) Effects of bandwidth misestimation, and performance of Vantage compared to using perfect bandwidth estimates.



(c) Comparison of Vantage with simpler retransmission schemes based on restricted versions of Vantage.

Figure 6: Sensitivity analysis experiments for Vantage.

simpler mechanisms such as reserving a fixed amount of bandwidth for enhancing the quality of the video frames affected by network impairments. The effectiveness of using Vantage can be attributed to three core design ideas: (1) optimizing bandwidth allocation across the real-time and retransmitted frames, (2) optimizing the choice of which frames to retransmit based on the population distribution, (3) and predicting the SSIM of the real-time and quality-enhancing frames for the optimization formulation. To understand the benefits of using Vantage over simpler designs, we conducted ablation studies comparing Vantage to restrained versions of Vantage.

Worst frame heuristic only considers the worst 30 frames for scheduling in each iteration, but lets Vantage perform bandwidth allocation based on the bitrate-SSIM curves to optimize the quality across the time-shifted delays. This is similar to a naïve solution that performs optimal bandwidth allocation, but selects frames for quality-enhancing retransmissions every 2 seconds according to a heuristic that picks a few of the worst frames. This approach works well for the two extreme viewing delays (real-time and 30 seconds of delay), but performs poorly for intermediate delays.

Unweighted optimization runs Vantage with equal weights for each frame, giving equal importance to real-time frames and retransmissions. This is similar to a slightly smarter heuristic that prefers contiguous segments for retransmission (since we penalize the selection of isolated frames in the optimization).

No SSIM prediction runs Vantage with a fixed bitrate-SSIM curve. As a comparison point for Vantage, this ablation demonstrates the importance of performing smart bandwidth allocation across individual frames based on the actual video quality instead of a simple heuristic based bandwidth allocation strategy.

The results of running Vantage and the restrained versions on the AT&T LTE trace and the talking heads video are shown in Figure 6c. We observe that Vantage performs significantly better than the restrained versions, demonstrating the benefits of each design feature Vantage's design.

6 RELATED WORK

Previous work related to video streaming including WebRTC [9], RTMP [8], MPEG-DASH [5], HLS [3] and SVC [35] are discussed in earlier sections. We discuss additional related work in this section.

Joint bitrate selection and transport. Salsify [18] couples encoding and transport in order to match available bandwidth for real-time communication. This improves the quality and delay for interactive video but does not consider the variety of delays at which viewers interact with a single video, which Vantage targets.

Downstream path. There has been a considerable amount of work in improving content delivery on the downstream path for viewers with heterogeneous network capabilities (e.g., [30, 38, 45, 46]). In contrast, Vantage focuses on improving quality of experience by mitigating network variability on the *upload path*. Targeting the upload path is critical since techniques for improving the download quality are rendered ineffective if the quality of the uploaded video was poor to begin with. This is especially relevant for SLVS platforms due to the high degree of variability of mobile networks over which streams are commonly broadcasted.

Prioritization in multimedia. Several previous works have developed transport protocols [17] and application-specific optimizations (e.g., 360-video [13, 20, 28, 29]) which prioritize multimedia to improve quality. Vantage uses a similar technique (of selective prioritization) to prioritize real-time frames over retransmissions.

7 CONCLUSION

We have designed and implemented Vantage, a live video upload system that explicitly accounts for the diverse time-shifted viewing delays common in social live video streaming platforms. Vantage balances available upload bandwidth between real-time frames and quality-enhancing retransmissions of previously impaired frames, resulting in high QoE for real-time and time-shifted viewing *simultaneously*. Compared to existing live video upload solutions tailored for either real-time or delayed viewing, Vantage improves the SQI-SSIM for time-shifted viewing by 19.9% on average, with only minor reductions in real-time or delayed quality. These results showcase the benefit of optimizing video upload to cater to the diverse time-shifted viewing nature of emerging live video applications.

ACKNOWLEDGEMENTS

This material is based upon work supported by the National Science Foundation Graduate Research Fellowship under Grant Nos. DGE-1745016 and DGE-1252522, the National Science Foundation Computer and Network Systems under Grant No. CNS-1565343, and the Facebook Communications and Networking Research award. CloudLab [33] resources were used in running experiments. We would also like to thank Keith Winstein for providing valuable feedback during the shepherding process.

REFERENCES

[1] Facebook Live. https://live.fb.com/. Last accessed 18 June 2018.

[2] Hangouts On Air with YouTube Live. https://support.google.com/youtube/answer/7083786?hl=en. Last accessed 4 July 2018.

[3] HTTP Live Streaming. https://developer.apple.com/streaming/. Last accessed 19 June 2018.

[4] More Ways To Connect with Friends in Facebook Live. https://newsroom.fb.com/news/2017/05/more-ways-to-connect-with-friends-in-facebook-live/. Last accessed 4 July 2018.

[5] MPEG-DASH standard. https://mpeg.chiariglione.org/standards/mpeg-a/mpeg-dash. Last accessed 23 June 2019.

[6] Open Broadcaster Software. https://obsproject.com/. Last accessed 19 June 2018.

[7] Periscope. https://www.pscp.tv/. Last accessed 19 June 2018.

[8] Real-Time Messaging Protocol (RTMP) Specification. https://www.adobe.com/devnet/rtmp.html. Last accessed 19 June 2018.

[9] WebRTC. https://webrtc.org/. Last accessed 19 June 2018.

[10] Xiph.org Test Media. https://media.xiph.org/. Last accessed 27 January 2019.

[11] Youtube-Live. https://www.youtube.com/channel/UC4R8DWoMoI7CAwX8_LjQHig. Last accessed 18 June 2018.

[12] Facebook Live video for News Feed (part 2). https://atscaleconference.com/videos/facebook-live-video-for-news-feed-part-2/. Last accessed 19 June 2018.

[13] Hamed Ahmadi, Omar Eltobgy, and Mohamed Hefeeda. 2017. Adaptive multicast streaming of virtual reality content to mobile users. In *Proceedings of the on Thematic Workshops of ACM Multimedia 2017*.

[14] Cisco. 2017. *Cisco Visual Networking Index: Forecast and Methodology, 2016-2021*. Technical Report 1465272001663118.

[15] Luca De Cicco, Gaetano Carlucci, and Saverio Mascolo. 2013. Experimental Investigation of the Google Congestion Control for Real-time Flows. In *Proceedings of the 2013 ACM SIGCOMM Workshop on Future Human-centric Multimedia Networking (FhMN '13)*. ACM, New York, NY, USA, 21–26. https://doi.org/10.1145/2491172.2491182

[16] Zhengfang Duanmu, Kai Zeng, Kede Ma, Abdul Rehman, and Zhou Wang. 2016. A quality-of-experience index for streaming video. *IEEE Journal of Selected Topics in Signal Processing* 11, 1 (2016), 154–166.

[17] Aiman Erbad and Charles Buck Krasic. 2012. Sender-side buffers and the case for multimedia adaptation. *Commun. ACM* 55, 12 (2012), 50–58.

[18] Sadjad Fouladi, John Emmons, Emre Orbay, Catherine Wu, Riad S. Wahby, and Keith Winstein. 2018. Salsify: Low-Latency Network Video through Tighter Integration between a Video Codec and a Transport Protocol. In *15th USENIX Symposium on Networked Systems Design and Implementation (NSDI 18)*.

[19] LLC Gurobi Optimization. Gurobi Optimizer Reference Manual. http://www.gurobi.com

[20] Jian He, Mubashir Adnan Qureshi, Lili Qiu, Jin Li, Feng Li, and Lei Han. 2018. Favor: Fine-Grained Video Rate Adaptation. In *Proceedings of the 9th ACM Multimedia Systems Conference (MMSys 18)*.

[21] Stefan Holmer, Mikhal Shemer, and Marco Paniconi. 2013. Handling Packet Loss in WebRTC. In *2013 20th IEEE International Conference on Image Processing (ICIP 13)*.

[22] Te-Yuan Huang, Ramesh Johari, Nick McKeown, Matthew Trunnell, and Mark Watson. 2014. A Buffer-based Approach to Rate Adaptation: Evidence from a Large Video Streaming Service. In *Proceedings of the 2014 ACM Conference on SIGCOMM (SIGCOMM '14)*. ACM, New York, NY, USA, 187–198. https://doi.org/10.1145/2619239.2626296

[23] Junchen Jiang, Vyas Sekar, Henry Milner, Davis Shepherd, Ion Stoica, and Hui Zhang. 2016. CFA: A Practical Prediction System for Video QoE Optimization. In *13th USENIX Symposium on Networked Systems Design and Implementation (NSDI 16)*.

[24] Junchen Jiang, Vyas Sekar, and Hui Zhang. 2012. Improving Fairness, Efficiency, and Stability in HTTP-based Adaptive Video Streaming with FESTIVE. In *Proceedings of the 8th International Conference on Emerging Networking Experiments and Technologies (CoNEXT 12)*.

[25] Junchen Jiang, Vyas Sekar, and Hui Zhang. 2014. Improving Fairness, Efficiency, and Stability in HTTP-Based Adaptive Video Streaming With Festive. *IEEE/ACM Trans. Netw.* 22, 1 (Feb. 2014), 326–340. https://doi.org/10.1109/TNET.2013.2291681

[26] Colin Levy and Ton Roosendaal. 2010. Sintel. In *ACM SIGGRAPH ASIA 2010 Computer Animation Festival (SA '10)*. ACM, New York, NY, USA, Article 82, 1 pages. https://doi.org/10.1145/1900264.1900346

[27] Zhenyu Li, Mohamed Ali Kaafar, Kave Salamatian, and Gaogang Xie. 2017. Characterizing and Modeling user Behavior in a Large-scale Mobile Live Streaming System. *IEEE Transactions on Circuits and Systems for Video Technology* 27, 12 (2017), 2675–2686.

[28] Xing Liu, Qingyang Xiao, Vijay Gopalakrishnan, Bo Han, Feng Qian, and Matteo Varvello. 2017. 360 Innovations for Panoramic Video Streaming. In *Proceedings of the 16th ACM Workshop on Hot Topics in Networks (HotNets 17)*.

[29] Kaixuan Long, Chencheng Ye, Ying Cui, and Zhi Liu. 2019. Optimal Multi-Quality Multicast for 360 Virtual Reality Video. *arXiv preprint arXiv:1901.02203* (2019).

[30] Hongzi Mao, Ravi Netravali, and Mohammad Alizadeh. 2017. Neural Adaptive Video Streaming with Pensieve. In *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*.

[31] Ravi Netravali, Anirudh Sivaraman, Somak Das, Ameesh Goyal, Keith Winstein, James Mickens, and Hari Balakrishnan. 2015. Mahimahi: Accurate Record-and-Replay for HTTP. In *2015 USENIX Annual Technical Conference (USENIX ATC 15)*.

[32] Reza Rejaie, Mark Handley, and Deborah Estrin. 2000. Layered quality adaptation for Internet video streaming. *IEEE Journal on Selected Areas in Communications* 18, 12 (2000), 2530–2543.

[33] Robert Ricci, Eric Eide, and The CloudLab Team. 2014. Introducing CloudLab: Scientific Infrastructure for Advancing Cloud Architectures and Applications. *USENIX ;login:* (2014).

[34] Luigi Rizzo. 1997. Dummynet: a Simple Approach to the Evaluation of Network Protocols. *ACM SIGCOMM Computer Communication Review (CCR)* (1997).

[35] Heiko Schwarz, Detlev Marpe, and Thomas Wiegand. 2007. Overview of the Scalable Video Coding Extension of the H.264/AVC Standard. *IEEE Transactions on Circuits and Systems for Video Technology* 17, 9 (2007), 1103–1120.

[36] Douglas Soo. Twitch Engineering: An Introduction and Overview. https://bit.ly/2JGR5yb. Last accessed 19 June 2018.

[37] John C Tang, Gina Venolia, and Kori M Inkpen. 2016. Meerkat and Periscope: I Stream, you Stream, Apps Stream for Live Streams. In *Proceedings of the 2016 CHI Conference on Human Factors in Computing Systems (CHI 16)*.

[38] Ming Tang, Lin Gao, Haitian Pang, Jianwei Huang, and Lifeng Sun. 2017. Optimizations and Economics of Crowdsourced Mobile Streaming. *IEEE Communications Magazine* 55, 4 (2017), 21–27.

[39] Luis Teixeira. 2011. Rate-distortion Analysis for H.264/AVC Video Statistics. In *Recent Advances on Video Coding*. InTech.

[40] Bolun Wang, Xinyi Zhang, Gang Wang, Haitao Zheng, and Ben Y Zhao. 2016. Anatomy of a Personalized Livestreaming System. In *Proceedings of the 2016 Internet Measurement Conference (IMC 16)*.

[41] Xiaodong Wang, Ye Tian, Rongheng Lan, Wen Yang, and Xinming Zhang. 2018. Beyond the Watching: Understanding Viewer Interactions in Crowdsourced Live Video Broadcasting Services. *IEEE Transactions on Circuits and Systems for Video Technology* (2018).

[42] Zhou Wang, Alan C Bovik, Hamid R Sheikh, and Eero P Simoncelli. 2004. Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing* 13, 4 (2004), 600–612.

[43] Stefan Winkler and Praveen Mohandas. 2008. The evolution of video quality measurement: from PSNR to hybrid metrics. *IEEE Transactions on Broadcasting* 54, 3 (2008), 660–668.

[44] Keith Winstein, Anirudh Sivaraman, and Hari Balakrishnan. 2013. Stochastic Forecasts Achieve High Throughput and Low Delay over Cellular Networks. In *Presented as part of the 10th USENIX Symposium on Networked Systems Design and Implementation (NSDI 13)*.

[45] X Rex Xu, Andrew C Myers, Hui Zhang, and Raj Yavatkar. 1997. Resilient Multicast Support for Continuous-Media Applications. In *Proceedings of the IEEE 7th International Workshop on Network and Operating System Support for Digital Audio and Video (NOSSDAV 97)*.

[46] Jongwon Yoon, Honghai Zhang, Suman Banerjee, and Sampath Rangarajan. 2012. MuVi: A Multicast Video Delivery Scheme for 4G Cellular Networks. In *Proceedings of the 18th Annual International Conference on Mobile Computing and Networking (Mobicom 12)*.

Note: These sections contain material that was not peer-reviewed.

APPENDIX A SLVS PLATFORM EXPERIMENTAL SETUP

In this section, we describe the details of our experiments from Section 3.1.4 in evaluating the support for time-shifted quality improvements in popular SLVS platforms.

Opportunities to improve QoE for time-shifted viewers come into play after the network has recovered from an impairment, where the video was encoded at a lower bitrate to prevent network saturation. To determine whether a platform takes time-shifted QoE into consideration, we initiate a live stream and let it run on an unimpaired network for 40 seconds and then momentarily throttle the bandwidth to 0 Mbps (no data can be transmitted) for 3 seconds. This is followed by 30 seconds of transmission over an unimpaired network. We emulate an impaired network using the `dummynet` [34] network emulator. Note that there is no impairment on the link between the live streaming platform's upload endpoint and the viewer. While our experiments may also contain variations that occurs in the upload path outside of the impairment we impose, we found the upload path to be fairly stable during all experiments.

We test the RTMP protocol using OBS [6] with settings recommended by the platform under test, and the WebRTC protocol using Google Chrome's WebRTC implementation. We compare the quality of the real-time and the archival versions of the video by comparing the video at the receiver to the prerecorded reference video that was uploaded.

APPENDIX B VIDEOS EVALUATED

(a) Talking Heads: Talking head video with a static background.

(b) City Panning: Panning motion with high detail.

(c) Animation: Animated sequence with varying amounts of motion.

Figure 7: Screenshots from the videos used in evaluation. Videos are drawn from the Xiph.org Test Media repository [10].

Figure 7 shows screenshots of the three different videos considered for evaluation, as described in Section 5.1.3.

APPENDIX C EFFECTS OF BANDWIDTH ESTIMATION ERROR

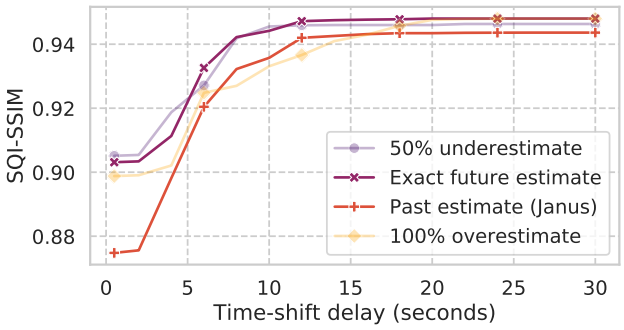


Figure 8: Comparison of Vantage with varying degrees of bandwidth estimation error for the Verizon LTE trace. This trace generally has very high bandwidth with only a brief period of impairment. In the case of underestimation, a lower bandwidth is allocated to the retransmissions, which results in better real-time performance. The higher real-time quality also translates to slightly better quality for delayed viewing as well. We observe that Vantage can still achieve significant improvements for delayed viewing even in the face of network variations, but at a slightly higher cost for the real-time quality in this case.

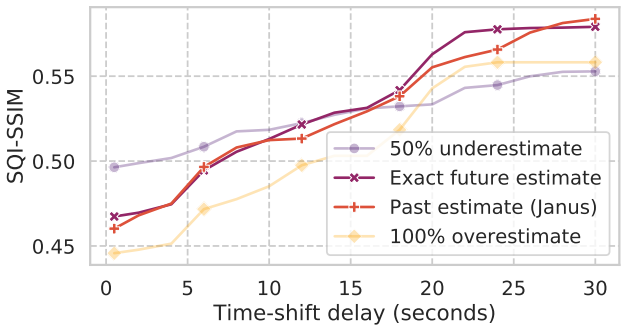


Figure 9: Comparison of Vantage with varying degrees of bandwidth estimation error for the TMobile UMTS trace. While the bandwidth in this trace is very low, it does not change very rapidly. The bandwidth is either a constant value or zero for extended periods of time. Hence, in this case, the performance of Vantage is very close to optimal. The low bandwidth causes significantly worse performace for the case of overestimated bandwidth, and we observe that it performs poorly across the entire delay range.