LBVC: Towards Low-bandwidth Video Chat on Smartphones

Xin Qi, Qing Yang, David T. Nguyen, Gang Zhou, Ge Peng
Department of Computer Science
College of William and Mary
Williamsburg, VA 23185, USA
{xqi, qyang, dnguyen, gzhou, gpeng}@cs.wm.edu

ABSTRACT

Video chat apps enable users to stay in touch with their family, friends and colleagues. However, they consume a lot of bandwidth and hence can quickly use up a monthly data plan quota, which is a high-cost resource on smartphones. In this paper, we propose LBVC (Low-bandwidth Video Chat), a user-guided vibration-aware frame rate adaption framework. LBVC takes a sender-receiver cooperative approach and reduces bandwidth usage as well as alleviates video quality degradation for video chat apps on smartphones. We implement LBVC on the Android platform and evaluate its performance on a series of experiments and user study with 21 pairs of subjects. Compared to the default solution, LBVC decreases bandwidth usage by 35% and at the same time maintains good video quality without introducing extra power consumption under typical video chat scenarios.

Categories and Subject Descriptors

C.4 [**Performance of Systems**]: Design Studies

General Terms

Measurement, Design, Performance

Keywords

Video Chat, Frame Rate Adaption, Frame Interpolation, Smartphones

1. INTRODUCTION

Nowadays various smartphone apps have been developed for people to keep in touch with their family, friends and colleagues. No matter whether apps of this kind have been commercialized (such as FaceTime, GoogleTalk and Skype) or not (such as CSipSimple, Linphone and SipDroid), almost all of them support video chat function. Through video chats, users not only hear people's voice, but also observe other contexts such as people's expressions and locations. As 4G service becomes prevalent, network speed is no longer a bottleneck of performing mobile video chats. According to recent reports, about 19% of Americans have contacted others using video chats [14] and it is expected that there will be 29 million smartphone video chat users in 2015 [3].

However, streaming a video chat is a data-intensive operation and results in high bandwidth usage. For example, the upload and download bandwidth requirements for low quality video chats over Skype are both 300Kbps [12]. Taking T-Mobile [4] as an example, its 500MB monthly 4G data plan, with the cost of $50 per phone, can support the use of such low quality video chats over Skype only for 1.85 hours. Although some carriers offer unlimited data plans, they come with significantly higher prices. For instance, the cost of T-Mobile's unlimited data plan is about 1.5 times higher than that of its limited data plan. Moreover, it has been reported that most carriers tend to only offer limited data plans [49], and going over the data limit means either higher cost or lower network speed [23]. Therefore, reducing the bandwidth usage of video chat apps on smartphones is imperative.

In this paper, we propose LBVC (Low-bandwidth Video Chat), a user-guided vibration-aware frame rate adaption framework. It reduces bandwidth usage as well as alleviates video quality degradation for video chat apps on smartphones. LBVC provides a user-friendly interface to help even non-technical users set an appropriate frame rate to save bandwidth with great facility. To be informative, the interface provides the estimated bandwidth usage and video quality degradation of each candidate frame rate compared to the default frame rate.

To save bandwidth and alleviate video quality degradation, LBVC introduces a vibration-aware sender-receiver cooperative approach. The sender adopts the lower user input frame rate to save bandwidth most of the time. It only adopts the default frame rate when it detects severe smartphone vibrations with respect to inertial sensors (accelerometer and gyroscope) readings. The receiver interpolates the 'missing' frames when the instant frame rate (calculated based on frame interval) is smaller than the default one. The purpose of the frame rate adaption at the sender is to keep the scene change between consecutive frames small in order to prevent strong artifacts from the frame interpolation.

It has been demonstrated that video streaming apps are among the most popular smartphone apps [26]. However, "*they consume much more bandwidth than other apps*" [18]. To lower the bandwidth requirement of video streaming,

MMSys '15, March 18 - 20, 2015, Portland, OR, USA

http://dx.doi.org/10.1145/2713168.2713180.

many video encoding techniques [8] [17] [45] [48] have been proposed. Rather than a new encoding technique, LBVC introduces a novel frame rate adaption framework that reduces the number of frames that need to be encoded by video chat apps on smartphones. Some works [23] [31] propose solutions that lower the bandwidth usage of downloading videos. They perform the trade-off between the quality of outgoing videos and bandwidth usage only at video sender. In contrast, LBVC reduces bandwidth usage more aggressively at video sender and does not guarantee the quality of outgoing videos. Instead, it utilizes extra techniques at both video sender (vibration-aware frame rate control) and receiver (frame interpolation) to guarantee the video quality at the receiver.

The main contributions of this paper are summarized as follows:

- We propose LBVC, a user-guided vibration-aware frame rate adaption framework for video chat apps on smartphones.
- We implement LBVC as a framework extension to the Android-based version of Linphone, a popular open-source video chat app for smartphones.
- We validate LBVC using real system experiments and a user study. The results demonstrate that LBVC saves bandwidth by 35% compared to the default solution and maintains good video quality.

The rest of the paper is organized as follows. We first present the experimental measurements that motivate our work. Then, the design and implementation of the LBVC framework are described. Next, we present the real system evaluation and user study. We finally discuss future work, elaborate existing works and conclude the paper.

2. MEASUREMENTS

One intuitive way to reduce the bandwidth usage of video chat apps on smartphones is to decrease frame rate. In this section, we answer the following two questions through measurements: (i) how does frame rate impact bandwidth usage and power consumption; (ii) how does frame rate impact video quality.

2.1 Measurement Platforms

In the measurements, we utilize two recent Android-based smartphones: Galaxy Nexus and Nexus 4. We choose Linphone [4], an open-source and lightweight smartphone VoIP (Voice over Internet Protocol) app, as the prototype software upon which we perform the measurements. Linphone is one of the most popular video chat apps in Android Market, with more than 100,000 downloads. It has support for different platforms, including Android devices, iPhone and desktop.

The Android version of Linphone is composed of a user interface, a multimedia library (called *Mediastreamer2*) and a library supporting SIP (Session Initiation Protocol) /RTP (Real-time Transport Protocol). *Mediastreamer2* is a native C++ library for recording, encoding/decoding and playing video and audio. We control the frame rate of video chats by configuring the corresponding parameter in the *Mediastreamer2* library. We keep the default configuration in SILK [11], the codec used by Linphone for audio encoding/decoding. We set H.264/AVC [17] as the codec for video encoding/decoding in Linphone. It utilizes both the intra-frame and inter-frame prediction techniques to compress video.

In H.264/AVC codec, the size of a compressed video depends on the rate control method adopted. Some rate control methods, such as Bitrate (BR) and Average Bitrate (ABR), fix the average encoding bitrate. Using these methods, the size of a compressed video is known in advance no matter what frame rate of the original video is. In contrast, other rate control methods, such as Constant Quantizer (CQ), Constant Rate Factor (CRF) and Lossless Mode, control quality instead of bitrate. Using these methods, the size of a compressed video varies with frame rates. Thus, this group of rate control methods allow us to reduce bandwidth usage through adopting lower frame rates. In this paper we choose CRF, since CRF achieves the best subjective visual quality without significant increase in video size [17]. We set the target quality (CRF parameter) to 22. From real world experience, this value is able to achieve satisfiable visual quality for most purposes [7].

2.2 Bandwidth Usage vs. Frame Rate

In this subsection, we investigate through measurement how does the frame rate reduction decrease the bandwidth usage? Since no data limit exists for smartphones when data is transmitted through a WiFi network, we only do the measurements over a cellular network.

We select 7 frame rates, which are 1, 2, 4, 8, 12, 16 and 20 fps (frames per second). The frame rates, ranging from $12 \sim 20$ fps, are the typical ones adopted by existing video chat apps on smartphones. For example, by default Linphone records video chats with 12 fps, while SipDroid records video chats with 20 fps. To be fair, we set the same values to all parameters, except for the frame rate, in all measurements. In each measurement, the Linphone app is compiled with one of the selected frame rates and installed on both smartphones. The video compression function in H.264/AVC codec is enabled.

To capture the network traffic, we utilize an app called *Shark for Root* [10] in Android Market. It is developed based on tcpdump [13], a widely used software for capturing network traffic. The captured network traffic is stored in *pcap* files, which is then analyzed using WireShark [15].

At each selected frame rate, 10 pairs of subjects perform 10 video chats, each of which lasts for 6 minutes. We choose 6 minutes as the video chat duration in the measurements, since it is recently reported as the average duration of mobile video chats [1]. During each video chat, two smartphones are separately held by two subjects who perform a typical video chat using Linphone. These two smartphones access Internet through the T-Mobile HSPA+ infrastructure. We measure the average bandwidth usage of the Galaxy Nexus in each video chat. The network traffic of the Nexus 4 is symmetric. During the measurements, we disable all other apps and services not required by Linphone and Shark for Root.

We utilize WireShark to filter the captured packets of the video chat app. These network packets include network control packets and data packets containing both audio and video data. Then, we calculate the average total bandwidth usage of the Galaxy Nexus during each video chat. We illustrate the box plot of the measured average bandwidth usage at the selected frame rates in Figure 1.

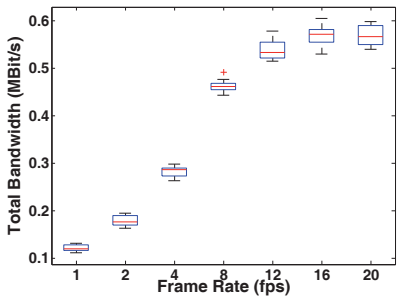

Figure 1: Total Bandwidth vs. Frame Rate on Galaxy Nexus.

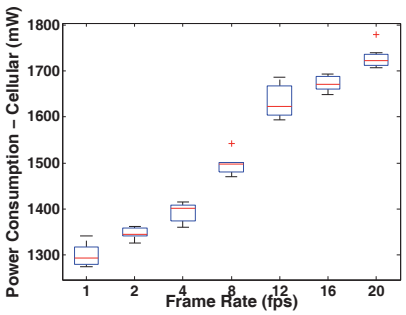

Figure 2: Power Consumption vs. Frame Rate on Galaxy Nexus in a Cellular Network.

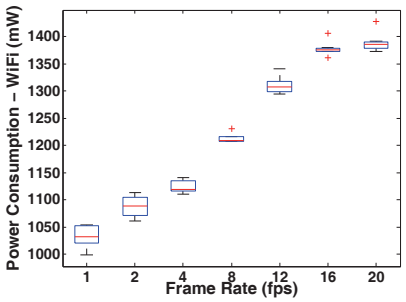

Figure 3: Power Consumption vs. Frame Rate on Galaxy Nexus in a WiFi Network.

In the figure, when the frame rates of the video chat are below 12 fps, we observe that the total bandwidth usage rapidly decreases as the frame rate decreases. For example, the total bandwidth usage at 12 fps is about 188% of that at 4 fps. Above 12 fps, the bandwidth usage increases slowly since video compression techniques begin taking effects.

The observations are obtained when video compression techniques in the codec (H.264/AVC) are involved. The observations indicate two facts below the typical frame rate range (12∼20 fps): (i) *the video compression techniques fail to efficiently compress video chats when frame rate is low;* (ii) *reducing frame rate is a feasible way to efficiently reduce bandwidth usage below the typical frame rate range adopted by existing video chat apps on smartphones.*

2.3 Power Consumption vs. Frame Rate

In this subsection, we investigate through measurement how does the frame rate reduction decrease the power consumption of performing video chats? Since the power constraint exists for smartphones no matter what type of network the data is transmitted through, we conduct the measurements over a public college WiFi network and the T-Mobile HSPA+ cellular network.

We use the Monsoon Power Monitor [5] to measure the power consumption of performing video chats with Linphone on Galaxy Nexus under different frame rates. We organize a similar set of experiments as the one in the previous subsection. In experiments, we disable Shark for Root and all other apps and services not required by Linphone. The average power consumption of each video chat is measured as follows.

We first measure the average power consumption when the smartphone's screen is on but no video chat is ongoing. Then, for each video chat, we measure its average power consumption, which is subtracted by the first measured power consumption to obtain the average power consumption of sampling, processing, encoding/decoding, transmitting/receiving and playing the video and audio.

We illustrate the box plot of the measured average power consumption in Figure 2 and 3. From the figures, we observe that the power consumption increases as the frame rate increases. It is because the smartphone has to process, encode/decode and transmit/receive more video frames at higher frame rates.

However, compared to the potential bandwidth usage reduction (5.24% ∼ 78.72% for the cellular network), the potential power consumption reduction (2.66% ∼ 24.72% for the cellular network and 0.5% ∼ 16.59% for the WiFi network) is much smaller. It is because the continuous multimedia stream keeps the smartphone networking component active.

These observations are consistent with existing works [40] and indicate that although reducing frame rate is able to reduce the power consumption of video chats, this power consumption reduction is not a major benefit.

In this subsection, we introduce the video quality metric we choose in this paper and illustrate the relationship between video quality and frame rate with this metric.

To quantify the video quality under different frame rates, we use a state-of-the-art non-reference objective video quality metric, TVM (Temporal Variation Metric), which is specifically designed for measuring video quality on mobile devices [22]. A non-reference metric is chosen, since no referring video exists at the receiver in a *live* chat.

TVM models the scene change occurred between consecutive frames in a video frame sequence. The range of a TVM score is from 0 to ∞. A lower score indicates that the scene changes between consecutive frames are jerky, and hence the video quality is low. A higher score indicates the opposites.

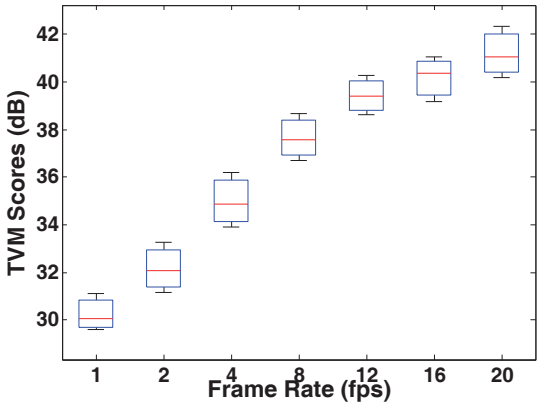

Figure 4: Video Quality vs Frame Rate.

2.4 Video Quality vs. Frame Rate

In the measurements, we use five 10-minute videos recorded with 30 frames per second by 5 different subjects under different backgrounds and light conditions. We adjust the

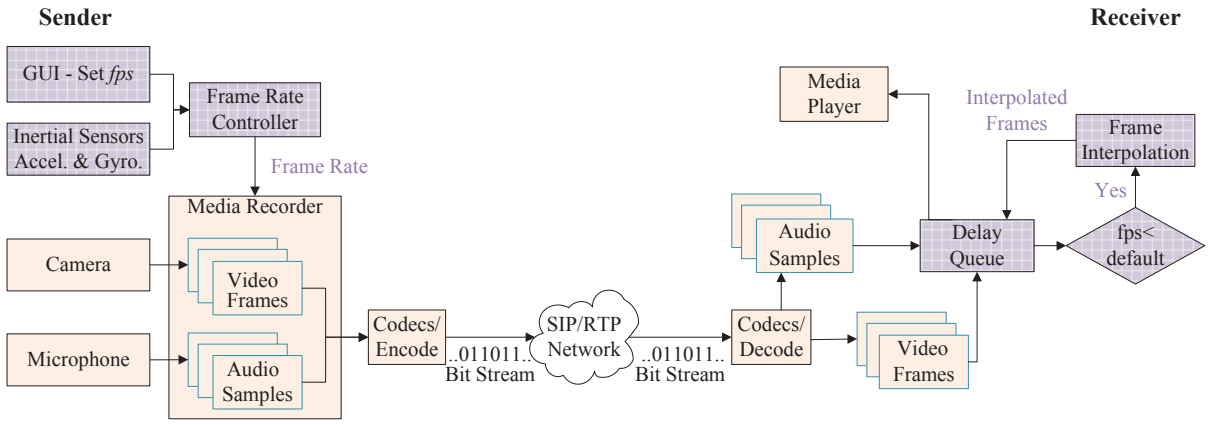

Figure 5: LBVC Architecture.

frame rate of each video off-line by ffmpeg [2] and calculate the TVM score for each video at each selected frame rate. We depict the box plot of all the scores in Figure 4.

From Figure 4, we observe that the objective video quality decreases as the frame rate decreases. For example, the TVM scores indicate that the mean square difference between consecutive frames at 4 fps increases by about 214% compared to that at 12 fps [22]. The larger the difference between consecutive frames, the jerkier the video. It is also reported that subjective video quality also decreases as frame rate decreases [39].

Therefore, we cannot simply reduce the frame rate of video chats to save bandwidth, because video quality degrades as frame rate decreases.

3. LBVC DESIGN AND IMPLEMENTATION

To save bandwidth usage of mobile video chats, we present the design and implementation of LBVC (Low-bandwidth Video Chat), a user-guided vibration-aware frame rate adaption framework.

3.1 LBVC Design

3.1.1 LBVC Architecture

Figure 5 depicts the architecture of LBVC with the common video chat app components in red and newly introduced components highlighted in purple grid. During a video chat, the communication is bi-direction and a smartphone is both a sender and receiver.

At the sender side of LBVC, users can set the frame rate for a video chat through a user-friendly interface. With respect to this input frame rate and on-line inertial sensor readings, a Frame Rate Controller adjusts the frame rate of the media recorder for video recording. To compress video frames and audio samples, the media recorder uses one of the existing codecs (e.g., SILK [11] and H.264/AVC [17] used in Linphone) to encode them. Then, the encoded data is transmitted over SIP/RTP networks.

At the receiver side of LBVC, the data packets received from the network is decoded into video frames and audio samples. To maintain video quality, the receiver adopts feasible frame interpolation technique to interpolate intermediate frames between any two received frames. The number of intermediate frames to be interpolated is chosen as the minimal number that makes the frame rate no smaller than the default frame rate.

A frame interpolation technique needs two frames as input. After a frame is decoded, the 'missing' (or intermediate) frames after it cannot be interpolated until the next frame arrives. To ensure the video frames being played at a constant speed, the former decoded frame cannot be played until the accomplishment of the frame interpolation. Thus, a delay should be added to the video player at the beginning of each video chat.

To ensure a constant speed of video playing as well as synchronize the audio and video, a Delay Queue adds a common delay to media player. The length of the common delay is determined based on the sender-side input frame rate, which the receiver obtains from the sender through SIP negotiation parameters [9]. Within the common delay, the Frame Interpolation Component interpolates the intermediate frames when the instant frame rate (calculated based on frame interval) is lower than the default one (e.g., 12 fps in Linphone). We note that the delay penalty (e.g., 260ms at 4 fps) is only paid once at the beginning of each video chat, and results in no desynchronization between audio and video during playing.

It is worth emphasizing that *LBVC is a generic design for typical video chat apps on smartphones, and is independent of lower layer components such as codecs and network protocols.*

3.1.2 Interface for Setting fps

The interface is designed to help users set an appropriate frame rate to save bandwidth for video chats. Particularly, it provides several candidate frame rates, which are smaller than or equal to the default frame rate of a video chat app. However, selecting a lower candidate frame rate not only reduces bandwidth usage, but also degrades the video quality on conversation partner side. Without any indication, even technical users may encounter difficulties in making a good trade-off between bandwidth usage and video quality.

Therefore, the interface also provides the estimated bandwidth usage and video quality degradation of each candidate frame rate compared to the default frame rate. With this knowledge, users are aware of the benefits and damages resulting from selecting a frame rate. Thus, they have more confidence when selecting an appropriate one with respect

to their needs. For example, a user who has sufficient data plan quota can select a higher candidate frame rate for better video quality. In contrast, a user who has limited data plan quota can select a lower candidate frame rate to reduce bandwidth usage.

3.1.3 Frame Rate Controller

Although the Frame Interpolation Component on the receiver side can rescue the video quality to some extent, even the most advanced frame interpolation technique may produce strong artifacts when the scene change between consecutive frames is large [21]. Therefore, we design a Frame Rate Controller that dynamically adapts frame rate so as to avoid large scene changes between consecutive frames.

The Frame Rate Controller utilizes the user input frame rate as the target frame rate. At run-time, it adopts the target frame rate to save bandwidth and only adopts the default frame rate, which is usually higher than the user input frame rate, when it detects large scene changes.

To detect large scene changes, we may directly use frame changes as the criteria. However, this method has a pitfall in that it requires multiple frames to detect large scene changes before making a decision. Waiting for multiple frames makes it unpractical because they cannot timely response to scene changes. For example, LBVC needs to wait 260ms for even one frame at 4 fps.

We also note that the typical causes of large scene changes between consecutive frames are severe smartphone vibrations. Therefore, to let Frame Rate Controller response in time, we design it to detect large scene changes by indirectly detecting severe smartphone vibrations. In LBVC, the Controller utilizes inertial sensors (accelerometer and gyroscope) readings to determine whether a smartphone vibration is severe or not during a video chat. In evaluation, we will demonstrate that this indirect method is still good enough for achieving acceptable video quality.

$$m_A = \sum_{t=start}^{end} [|\partial A_x(t)| + |\partial A_y(t)| + |\partial A_z(t)|] \quad (1)$$

$$m_G = \sum_{t=start}^{end} [|\partial G_x(t)| + |\partial G_y(t)| + |\partial G_z(t)|] \quad (2)$$

The Frame Rate Controller utilizes metrics based on L_1 norm of sensor reading changes [42] to quantify the smartphone vibration (See Equation 1 and 2). In the equations, $\partial A_x(t)$, $\partial A_y(t)$ and $\partial A_z(t)$ are the acceleration changes in three measured dimensions; while $\partial G_x(t)$, $\partial G_y(t)$ and $\partial G_z(t)$ are the angular velocity changes in three measured dimensions. In addition, *start* and *end* are the starting and ending time stamps of each measuring period. L_1 norm is adopted because it rigorously captures the smartphone vibrations signaled by sensor reading changes in any measured dimension.

In LBVC, the sampling rate of the sensors is set as 50 Hz and the measuring period is set as 60 milliseconds, which is small enough to quickly reflect the vibrations in real-time. If the value of either m_A or m_G is larger than a threshold, the smartphone vibration is considered *severe*. Otherwise, the vibration is considered *light*.

Moreover, the degree of vibration the Frame Interpolation Component can tolerate also depends on the input frame rate. For instance, if the input frame rate is set as a large value (such as 8 fps), it may generate intermediate frames without any artifacts under typical smartphone vibrations. However, if the input frame rate is set as a small value (such as 1 fps), it may generate intermediate frames with strong artifacts due to typical smartphone vibrations. Therefore, the vibration threshold values for different frame rates are different and should be determined experimentally by app developers with the objective of preventing obvious artifacts.

3.1.4 Frame Interpolation Algorithm

To select a proper frame interpolation algorithm for LBVC, we first investigate two optical-flow [21] [43] based algorithms. However, we find that their execution time is long. Take the algorithm in [43] as an example, we run it on a laptop which has even more advanced hardware configurations (Intel Core i7-2760QM CPU @ 2.4GHz ×8 and a memory of 12 Gigabytes) than recent smartphones such as Nexus 4. The input frames have similar resolution to the ones in smartphone video chats. Based on the timing records, the algorithm takes 825 seconds on average to interpolate one frame. A comprehensive survey of the execution time of various optical-flow based methods are listed in [6].

Since the optical-flow based algorithms are not practical in LBVC, we turn to the cross dissolve algorithm [32]. The cross dissolve algorithm is simple, since it only involves adding two frames and multiplying a frame and a constant. Moreover, it has been mathematically proved that "*given two images of a scene with small motion between them, a cross dissolve produces a sequence in which the edges move smoothly.*" [32]. In LBVC, the Frame Rate Controller on the sender is designed to prevent large scene changes between consecutive frames.

An example of the input and output of the cross dissolve algorithm is depicted in Figure 6. The time interval between the two input frames is 300 ms. As indicated by the equation in Figure 6, the intermediate frames calculated earlier obtain larger weights for the first input frame, while the ones calculated later obtain larger weights for the second input frame. Although the algorithm cannot generate the exact intermediate frames as the original ones, it is able to smooth the transition between the two input frames to some extent. In addition, artifacts such as blurring are alleviated when the interpolated frames are displayed in a short time interval (e.g. 300 ms in Figure 6).

3.2 LBVC Implementation

We implement LBVC as an extension to Linphone, a widely used open-source video chat app. Aside from Linphone, we also investigate the potential of modifying other popular open-source video chat apps, such as SipDroid and CSipSimple. We choose Linphone [4] as the base of our implementation because its software structure allows us to focus on the implementation of the video processing functionalities.

We implement the interface for user setting *fps* with Android SDK 4.2 in Java and integrate it to the Android version of Linphone. An example of the interface is shown in Figure 7. The app developers can add more candidate frame rates when they have the associated information.

The whole video streaming procedure in Linphone is implemented as filter graphs in the native *Mediastreamer2* library. Each filter in a graph is in charge of one task such as video capturing, encoding, or displaying. When a video stream starts, a thread called *Ticker* is scheduled to wake

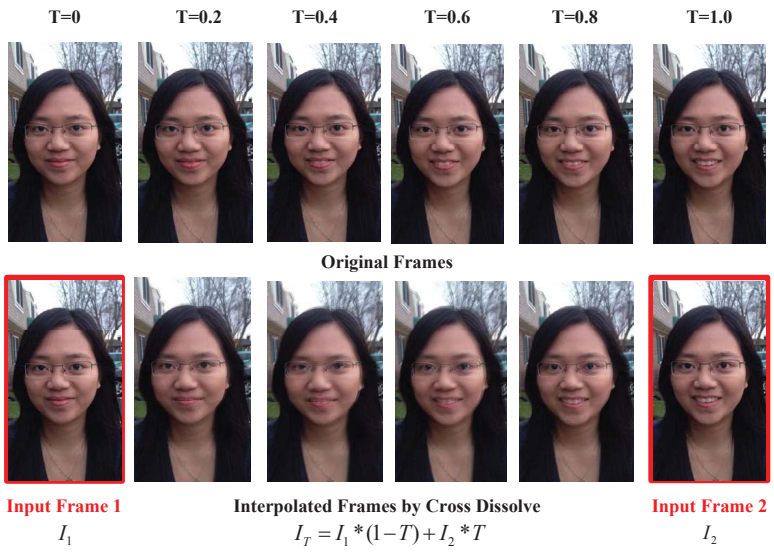

Figure 6: Cross Dissolve Example.

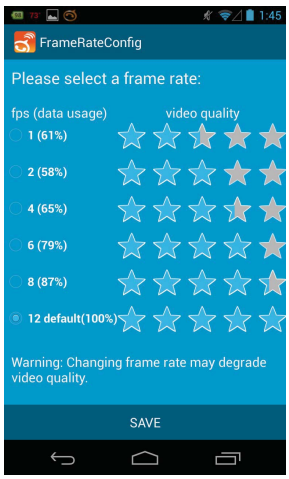

Figure 7: The Interface.

up every 10ms and executes the filter graphs. We implement the Frame Rate Controller as a component at the beginning of the Ticker thread. It collects the accelerometer and gyroscope readings through the Android NDK sensor library API. For every 60ms, if the calculated vibration metrics is larger than a specific threshold, the default frame rate (12 fps) is set for the video capturing and encoding filters; otherwise, the user-specified lower frame rate is set.

We implement the Frame Interpolation Component as a new filter between the video decoding and displaying filters. It utilizes the two most recently decoded frames as the input to the cross dissolve algorithm and calculates the intermediate frames. All the frames are put in a queue and assigned proper time stamps including the common delay for displaying. For each frame in the queue, it is not sent to the displaying filter unless its time stamp is not smaller than the current system time. The audio playing filter utilizes a similar queue to add the common delay for audio playing.

4. EVALUATION

In this section, we evaluate LBVC with a series of experiments and answer the following two questions: (1) How does LBVC impact the bandwidth usage and power consumption under typical video chat scenarios and mobility cases? (2) How does LBVC impact the objective and subjective video quality?

4.1 LBVC Configuration

Frame Rate (fps)	1	2	4	6	8
m_A THRs (m/s^2)	2.0	3.0	4.0	4.5	6.0
m_G THRs (rad/s)	2.5	3.0	3.2	3.5	4.0
Common Delay (ms)	1100	620	260	240	200

Table 1: Selected Frame Rates and Their Associated Parameters (Thresholds and Delay).

In the experiments, we use the implementation of the LBVC extension to Linphone. We select 5 typical frame rates that are lower than the default 12 fps, and summarize them in Table 1. For each selected frame rate, we list the thresholds for the Frame Rate Controller to adapt frame rate and common delay added to synchronize the audio and video (refer to the LBVC Architecture subsection). These values are determined empirically. Particularly, the thresholds are the minimal values that can prevent obvious frame interpolation artifacts resulting from smartphone vibrations. The delay is the upper bound of the frame interval observed from extensive experiments.

4.2 Performance Under Typical Scenarios

In this subsection, we evaluate the performance impact of LBVC at the selected frame rates in Table 1. We recruit the same 10 pairs of subjects in measurement section. Each pair performs video chats at all selected frame rates for bandwidth and power measurements on a Galaxy Nexus and Nexus 4. Each video chat lasts for 6 minutes. The subjects are instructed to perform video chats as usual under typical video chat scenarios such as sitting and standing. There are no other physical constraints imposed on the subjects. During all video chats, we disable all unnecessary apps, services, and radios on smartphones.

We perform bandwidth measurements over the T-Mobile HSPA+ cellular network, and power measurements over a public college WiFi network and the T-Mobile HSPA+ cellular network. We separate the video chats for bandwidth measurements from those for power measurements, since the network traffic capturing software also consumes power.

4.2.1 Bandwidth Usage

In the experiments, we utilize *Shark for Root* to record network traffic and WireShark to analyze bandwidth usage on a server.

Figure 8 illustrates the measured total bandwidth usage of uploading and downloading the video chats on Galaxy Nexus. The red line depicts the average total bandwidth usage at the default 12 fps with LBVC disabled. The blue bars illustrate the average total bandwidth usage at the selected frame rates with LBVC enabled.

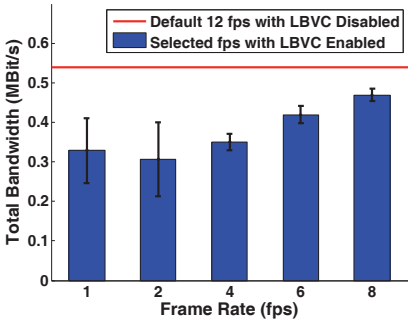

Figure 8: Bandwidth Usage vs. Frame Rate on Galaxy Nexus.

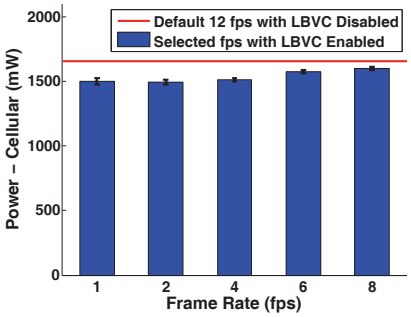

Figure 9: Power Consumption vs. Frame Rate in a Cellular Network on Galaxy Nexus.

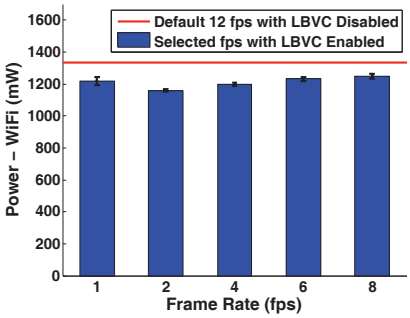

Figure 10: Power Consumption vs. Frame Rate in a WiFi Network on Galaxy Nexus.

In the figure, we observe that the average total bandwidth usage with LBVC enabled at 1 fps is even higher than that at 2 fps. This is due to the small thresholds used for frame rate adaption at 1 fps. Small thresholds allow some typical smartphone vibrations to trigger switches from lower input frame rate to higher default frame rate, resulting in larger bandwidth usage. At 2 fps, the larger thresholds result in less switches to the default frame rate, and hence the bandwidth usage with LBVC enabled is smaller.

Moreover, when LBVC is enabled, we observe that the bandwidth usage variances at 1 and 2 fps are larger than those at higher frame rates. It is because that the thresholds at 1 and 2 fps are smaller than those at higher frame rates. Smaller thresholds allow smartphone vibrations to trigger more switches between the input frame rate and the default frame rate. The unstable runtime frame rate results in large bandwidth usage variances. However, as the frame rate increases, increased thresholds result in smaller bandwidth usage variances.

Frame Rate (fps)	1	2	4	6	8
Galaxy Nexus (%)	39.1	43.2	35.2	22.3	13.0
Nexus 4 (%)	38.5	41.9	35.4	21.2	13.3

Table 2: Average Bandwidth Savings vs. Frame Rates on Galaxy Nexus and Nexus 4 (Compared to the Default 12 fps with LBVC Disabled).

We summarize the average bandwidth savings in Table 2. The average bandwidth savings for the two smartphones are similar because the communication is symmetric. The results demonstrate that LBVC is able to reduce the average bandwidth usage of video chats by up to 43.2% under typical video chat scenarios. Additionally, lower bandwidth usage at clients reduces the traffic flow competition at the RTP server, and consequently may result in less packet loss and jitter.

4.2.2 Power Consumption

In the experiments, we utilize the Monsoon Power Monitor [5] to measure the average power consumption of performing each video chat on Galaxy Nexus. However, the power output interface of the monitor cannot be connected to the battery pins of Nexus 4. Due to this hardware constraint, we use the software PowerTutor [50] to measure the average power consumption of performing each video chat on Nexus 4. In general, the measured power consumption includes the power for sampling, processing, encoding/decoding, transmitting/receiving and playing the video and audio. When LBVC is enabled, it also includes the power of the sensors and frame interpolation.

Figure 9 and 10 summarize the measured average power consumption of all the video chats over the cellular and WiFi networks on Galaxy Nexus. The red lines depict the average power consumption at the default 12 fps with LBVC disabled. The blue bars illustrate the average power consumption at the selected frame rates with LBVC enabled.

In the figures, we observe that the average power consumption at 1 fps is larger than that at 2 fps. Higher power consumption at 1 fps not only results from the frequent adaption to the default frame rate, but also results from more frame interpolation operations performed at 1 fps.

In addition, we observe that the power consumption variance with LBVC enabled at 1 fps is larger than those at higher frame rates. This is due to the unstable runtime frame rate. However, this variance is not significant compared to the total power consumption.

Frame Rate (fps)	1	2	4	6	8
Gal.N.-WiFi(%)	8.9	13.2	10.4	7.8	6.6
Gal.N.-Cell(%)	9.7	10.1	8.8	5.2	3.6
Nexus4-WiFi(%)	10.1	14.0	11.2	8.6	7.4
Nexus4-Cell(%)	11.9	13.6	10.3	8.7	5.1

Table 3: Average Power Savings vs. Frame Rates on Galaxy Nexus and Nexus 4 (Compared to the Default 12 fps with LBVC Disabled).

The average power savings are summarized in Table 3. From the table, we learn that although LBVC does not save much power, the power saving resulting from frame rate reduction is large enough to offset the power consumption of sensors and frame interpolation. Therefore, LBVC does not introduce extra power overhead under typical video chat scenarios.

4.3 Performance in Mobility Cases

Since the Frame Rate Controller uses the inertial sensors to adapt frame rate, the frame rate adaption is affected by subjects' mobility. In different mobility cases, the amount of time during which LBVC works at the input frame rate is different. Consequently, this affects the bandwidth savings.

In the previous section, we conducted experiments to demonstrate the bandwidth and power savings of LBVC while the subjects are stationary, such as sitting and standing. In this section, we conduct the same group of experiments in the other two mobility cases, walking, a representative of low mobility, and sitting in a moving vehicle, a representative of high mobility.

We summarize the percentages of time for LBVC to work at different input frame rates in all the three mobility cases in Figure 11. Since a higher input frame rate has a larger threshold for frame rate adaption, we observe that as the input frame rate increases, the percentage of time for LBVC to operate at that frame rate also increases.

In Figure 11, we also observe that the stationary case has the largest percentage of time for LBVC to work at user input frame rates among all the cases. It is because there are more smartphone vibrations in the other two cases. In order to compensate the increased smartphone vibrations and rescue video quality, the Frame Rate Controller has to frequently adapt the instant frame rate to the default one. In the stationary case, LBVC has the highest chance to save bandwidth.

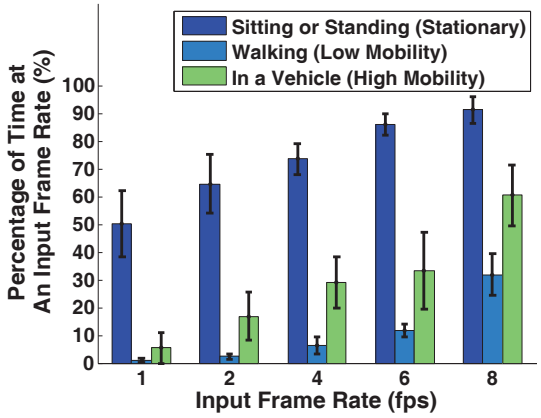

Figure 11: Percentage of Time for LBVC to Work at Different Input Frame Rates in Different Mobility Cases.

Additionally, we observe that LBVC has the lowest chance to work at input frame rates to save bandwidth in the walking case. It is because that the subjects do not control their walking speed and gestures during video chats when they are walking. Thus, in walking case, frequent speed changes and slight jolts cause LBVC operating at the default frame rate most of time.

In the walking case, although a subject's head does not have a significant relative movement in the screen during a video chat, the background of the video chat continuously changes. However, the fast background change is successfully compensated thanks to LBVC's frequent adaption to the default frame rate.

Finally, we observe that LBVC operates at input frame rates for more time in the vehicle case than it does in the walking case. It is because that the vehicles are usually operated at almost constant speed near the speed limits on the roads. At almost constant speed, acceleration changes are not large enough to trigger the switch to the default frame rate. However, different traffic conditions result in different speed controls. The variances of the percentages in the vehicle case are large.

In a vehicle, a subject's head usually has a significant relative movement in the screen when the vehicle changes speed during braking, accelerating or hitting a speed bump. In these scenarios, LBVC is triggered by the inertial sensors to adapt the frame rate in order to compensate the relative movement.

We summarize the bandwidth savings in Table 4. Although the bandwidth savings in these two cases are smaller than those under the typical scenarios, the extra consumed bandwidth compensates the frequent smartphone vibrations in order to alleviate video quality degradation. Since it is hard to run the power monitor while the subjects are walking or sitting in a vehicle and power saving is not a major benefit of LBVC, we do not perform power measurements in these two cases.

Frame Rate (fps)	1	2	4	6	8
Walking Case					
Galaxy Nexus (%)	0.7	1.6	3.0	3.1	4.5
Nexus 4 (%)	0.7	1.5	3.0	3.2	4.4
Vehicle Case					
Galaxy Nexus (%)	4.2	11.2	13.9	8.6	8.6
Nexus 4 (%)	4.4	11.2	13.10	8.8	8.6

Table 4: Average Bandwidth Savings vs. Frame Rates on Galaxy Nexus and Nexus 4 (Compared to the Default 12 fps with LBVC Disabled) in the Two Mobility Cases.

4.4 Video Quality

We perform a user study to evaluate the impact of LBVC on video quality and demonstrate that LBVC can alleviate the video quality degradation resulting from frame rate reduction. In the study, we recruit 21 different pairs of subjects (42 subjects in total) to perform video chats when LBVC is either disabled or enabled under each selected frame rate. Half of the subjects major in Computer Science, while the others major in varying areas, including Physics, Applied Science, Mathematics, Nursing, Education and Rhythmic Gymnastic. Each video chat lasts for 6 minutes. We collect both subjective and objective scores of each video chat.

To collect subjective scores, we carry out a Double Stimulus Continuous Quality Evaluation (DSCQE) [16], in which each subject rates a video chat by comparing it to a reference video. The scores are from 0 to 100. An intuitive explanation for the relationship between the video quality and score is as follows: very annoying (0-20); annoying (21-40); fair (41-60); good (61-80); excellent (81-100).

In each round of the DSCQE, we first ask each subject pair to perform and score a video chat under the default 12 fps with LBVC disabled and use this video chat as the reference one. Second, we configure the Linphone using a randomly chosen frame rate from Table 1 with LBVC either enabled or disabled. Then, each subject pair is instructed to perform and score a video chat under this configuration. During each video chat, the subjects are also instructed to vibrate the smartphone to simulate different mobility cases. We repeat the second step until all the selected frame rates with LBVC either enabled or disabled have been traversed by each subject pair. In addition, we also query them whether

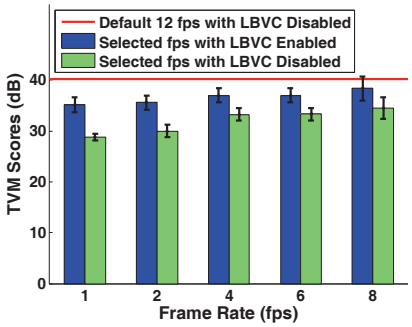

Figure 12: TVM Scores vs. Frame Rate. Error Bars indicate the Standard Deviations.

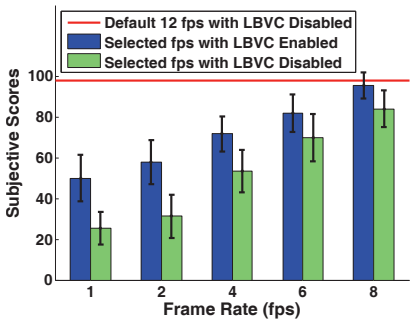

Figure 13: Subjective Scores vs. Frame Rate. Error Bars indicate the Standard Deviations.

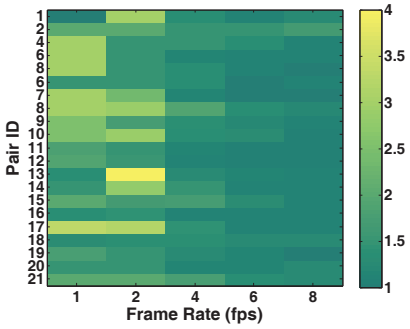

Figure 14: The Ratio of the Subjective Score with LBVC Enabled to the Subjective Score with LBVC Disabled for Each Pair.

the common delay affects the subjects' video chat experience or not.

To collect objective scores, we implement a recorder in Linphone to automatically record all video chats during the DSCQE. As in the measurement section, we use the Temporal Variation Metric (TVM) [22] to measure the objective video quality. We analyze the recorded videos by MATLAB on a PC to obtain the TVM scores.

The objective and subjective scores are depicted in Figure 12 and 13. In these two figures, the red lines depict the average score at the default 12 fps of Linphone with LBVC disabled. The blue bars illustrate the average scores at the selected frame rates with LBVC enabled, and the green bars illustrate the average scores at the selected frame rates with LBVC disabled.

Objective Scores Figure 12 demonstrates that LBVC is able to alleviate the objective video quality degradation resulting from frame rate reduction. For instance, the average TVM scores at 4 fps indicate that the mean square difference between consecutive frames with LBVC enabled is decreased by 58% compared to that with LBVC disabled. This significant decrease in the mean square difference between consecutive frames results in smoother videos.

Subjective Scores Figure 13 summarizes the subjective scores collected from the study. This figure demonstrates that LBVC is able to alleviate the perceptual video quality degradation resulting from frame rate reduction. For instance, the average subjective score at 4 fps with LBVC enabled is improved by 25.3% compared to the one with LBVC disabled. The score is even higher than that at 6 fps with LBVC disabled.

When LBVC is enabled, we also notice that the relationship between subjective score and frame rate is different from that between objective score and frame rate. There are two possible reasons. First, although the cross dissolve algorithm interpolates intermediate frames to smooth the video chats, the artifacts in the interpolated frames may also impair the subjects' perceptual experience. The lower the frame rate, the larger the scene change between consecutive frames, which consequently results in more artifacts in the interpolated frames. Second, the frequent changes between the default frame rate and input frame rate at lower frame rates (e.g., 1 and 2 fps) also impair the subjects' perceptual experience. In contrast, the TVM only measures the video jerkiness but not the aforementioned artifacts. However, Figure 13 demonstrates that the subjective perceptual experience is improved as a whole, although the artifacts offset some improvements.

Notwithstanding the aforementioned limitation, we still choose TVM for two reasons. First, very few non-reference metrics are designed for measuring video quality with different frame rates. TVM is the newest metric of this sort and specifically designed for mobile devices. Second, TVM scores are accurate enough to demonstrate the effective alleviation of video quality degradation.

Figure 14 is a heat map of the ratio of the subjective score with LBVC enabled and the one with LBVC disabled under each frame rate. We calculate the score of each pair as the average of the two subjects' scores. We observe that LBVC rarely worsens the video quality, and is able to alleviate the video quality degradation for most of the time.

From both Figure 13 and Figure 14, we observe that the video quality improvements are larger at 1 and 2 fps than those at other frame rates. It is because the extreme video jerkiness at 1 and 2 fps accentuates the smoothness brought by frame interpolation and frame rate control.

Finally, almost all the subjects claim that the impacts of the common delay during each video chat are acceptable and even negligible when video chats are performed at 4 frames per second and above. They also claim that there is no obvious desynchronization between audio and video during video chats.

From Table 2 and 3, and Figure 13, we notice that at 4 fps, *LBVC achieves about 35% bandwidth saving while still maintaining good video quality without introducing extra power consumption under typical video chat scenarios.*

5. DISCUSSION AND FUTURE WORK

In this section, we discuss the assumptions and limitations of LBVC. Then, we will discuss how to further improve LBVC in the future.

First, in the design, LBVC utilizes the inertial sensor readings to dynamically adapt the frame rate. One shortcoming of this approach is being unaware of the object movements in videos. For example, if the user's head moves fast, but the handheld smartphone does not, this approach may not be able to instantly adapt the frame rate in order to compensate for the fast head movement. However, since user's hand and head are connected through user's body, it is the usual case that user's head moves fast and hands also move

significantly. From this perspective, we do not observe such shortcoming frequently.

Second, we note that LBVC may miss some chance of bandwidth savings when users are sitting in a moving vehicle. Frequent vehicle speed changes make the accelerometer readings highly dynamic and frequently trigger the switch to higher default frame rate. To solve this problem, LBVC may first use sensors, such as accelerometer and GPS, to detect whether a user is sitting in a vehicle or not. With the contextual knowledge of being in a vehicle, LBVC then only uses the gyroscope to sense smartphone vibrations. We will extend LBVC to address this special case in the future.

Third, in the design, LBVC switches to the default frame rate when it detects severe vibrations with respect to the thresholds of input frame rate. The principle of this design is to provide enough quality for users under severe vibrations. However, LBVC can further reduce bandwidth through a more fine-grained frame rate adaption approach. In the approach, when a severe vibration is detected, LBVC does not necessarily switch to the default frame rate, which is higher than other candidate frame rates. Instead, it can increase frame rate to a candidate, which is high enough for frame interpolation to compensate the current vibration. With this approach, LBVC has much less chance to switch to the default frame rate, and in consequence has more chance to reduce bandwidth usage. Meantime, it may worsen the overall video quality. Thus, to explore this alternative approach, we should also investigate its impacts on video quality. We will leave these works in future.

Finally, in Section 4.1, we empirically set up the threshold values in such a way that they can prevent obvious frame interpolation artifacts resulting from smartphone vibrations. However, we are aware that different users may have different subjective criteria for whether interpolation artifacts are obvious or not. In future, instead of pinning down the thresholds manually for all users, we will explore the possibility of building a model that describes the relationship between frame rate and threshold values. Once the model is abstracted, it can be concreted for different users through learning model parameters.

6. RELATED WORK

The initial idea of LBVC is presented in a poster [41]. We further improve our idea from 5 aspects: (i) we design a user-friendly interface to help users with varying video quality requirements set an appropriate frame rate to save bandwidth with great confidences; (ii) we investigate the relationship between power consumption and frame rate, and the impact of our design on power consumption; (iii) we investigate bandwidth savings of our design in other mobility cases such as walking and sitting in a vehicle; (iv) we organize a user study and collect subjective scores to investigate whether our design can maintain subjects' perceptual experience; (v) we also investigate other frame interpolation algorithms, such as the optical-flow based algorithm in [43].

In this section, we articulate how LBVC is different from existing works of video streaming bandwidth reduction, video bitrate adaption, video streaming power reduction and image interpolation.

Video Streaming Bandwidth Reduction. To lower the bandwidth usage of streaming videos, many video compression techniques [8] [17] [25] [29] [45] have been proposed. VP8 [8] and H.264 [17] utilize motion compensation technique that exploits the temporal correlation among frames to reduce video sizes. SaVE [25] and SenseCoding [29] utilize accelerometers to simplify existing motion estimation methods. In addition to the aforementioned video compression techniques, existing works in compress sensing such as [37] and context-aware compression [19] are also related.

In general, video compression techniques exploit various information from video frames to reduce the number of necessary bits to encode them. In contrast, LBVC is designed to reduce the number of frames to be encoded, which is equivalent to reducing the size of the input to video compression techniques. Thus, LBVC and video compression are orthogonal.

Video Bitrate Adaption. Video bitrate adaption techniques take into account various information, such as networking conditions [24] [33] [44], video contents [35] and user preferences [46], to control video streaming bitrate in order to maximize video utility (or video quality) for video consumer [27]. Compared to these quality-oriented video bitrate adaption techniques, LBVC adjusts video chat bitrate to help user reduce their data plan quota usage and maintain satisfiable video quality.

Some recent works, such as QAVA [23]and TUBE [28], also propose techniques to help user economically use their data plan quota for video streaming. LBVC is different from these works from two perspectives. First, LBVC is designed to reduce the bandwidth usage of streaming *live* and *interactive* video chats instead of downloading *existing* videos. Second, QAVA and TUBE optimally perform the trade-off between video quality and bitrate (or bandwidth usage) at video sender in order to save date usage as well as guarantee the quality of outgoing videos. In contrast, LBVC reduces video bitrate more aggressively (through frame rate adaption) at video sender and does not guarantee the quality of outgoing videos. Instead, it utilizes extra techniques at both video sender (vibration-aware frame rate control) and receiver (frame interpolation) to guarantee the video quality at the receiver. In short, LBVC has more capacity to reduce bandwidth usage of video chats since it has more guarantees for video quality.

Video Streaming Power Reduction. In [38], an integrated power management method for video streaming is proposed. It achieves architectural power optimizations on CPU, register, and memory. It also includes an OS power-saving mechanism - dynamic voltage scaling, and adaptive middleware for video transcoding and network traffic regulation. Empirical measurements performed in [36] demonstrate that the chunk-based video streaming protocols are the most power efficient for downloading videos on mobile devices, since networking components on mobile devices have more chance to sleep. The method in [30] shortens the active duration of WiFi after each video prefetchings and cuts off unnecessary video pre-fetchings with respect to users' video viewing preferences. GreenTube [34] optimizes power consumption for mobile video downloading by judiciously scheduling downloading activities to minimize unnecessary active period of 3G/4G radio.

In contrast, LBVC is proposed to mainly reduce bandwidth usage of video chats streaming on smartphones, although it has the capacity of, but not significantly, reducing power consumption of video chats.

Image Interpolation. Exploring Photobios [32] also utilizes the cross dissolve algorithm to interpolate intermediate frames between two well-aligned images with the purpose of creating a face animation with smooth face transitions. InterviewVideo [20] provides a tool for automatically placing interpolated frames to make smooth transitions in interview videos. Some methods such as [21] and [43] propose optical-flow based warping algorithms to interpolate the frames. However, as we report in Section 3.1.4, warping algorithms usually involves complex operations, such as image derivation and matrix multiplication. Thus, they are more computationally intensive than the cross dissolve algorithm [32] adopted in LBVC implementation. Some works such as [47] utilize frame interpolation techniques to make up the lost frames in video transmission. Although LBVC also makes up frames, those frames are *deliberately dropped* (*not passively lost* in [47]) to save bandwidth.

7. CONCLUSION

How to appropriately balance the mobile video quality and bandwidth usage is a relevant but hard problem. Particularly, we attempt to address the problem in the scenario of video chats by proposing LBVC. LBVC adapts the video frame rate with respect to the smartphone vibrations at the sender side and interpolates the '*missing*' frames at the receiver side. Through real system experiments and a user study, we demonstrate that LBVC saves bandwidth by up to 43% under typical video chat scenarios at the expense of limited video quality degradation.

8. ACKNOWLEDGMENTS

The authors would like to express their gratitude to the reviewers who provided insightful comments and suggestions to improve the paper. The authors also wish to extend their thanks to the subjects, who participated in the experiments and user study. This work was supported in part by U.S. National Science Foundation under grant CNS-1250180 and CNS-1253506 (CAREER).

9. REFERENCES

[1] Average duration of mobile video chat. `http://goo.gl/ow9uDT`.
[2] ffmpeg. `http://www.ffmpeg.org/`.
[3] Juniper research report. `http://goo.gl/F5qvGz`.
[4] Linphone. `www.linphone.org`.
[5] Monsoon power monitor. `http://goo.gl/spM1pt`.
[6] Optical-flow execution time. `http://goo.gl/vcS1px`.
[7] Rate control in h.264. `http://goo.gl/7pfUp8`.
[8] Rfc6386 - vp8 data format and decoding guide. `http://tools.ietf.org/html/rfc6386`.
[9] Session description protocol. `http://goo.gl/fcphyP`.
[10] Shark for root. `http://goo.gl/IRCUVw`.
[11] Silk. `http://dev.skype.com/silk`.
[12] Skype bandwidth requirement. `http://goo.gl/apzY9s`.
[13] Tcpdump. `http://www.tcpdump.org`.
[14] Us video call statistics. `http://goo.gl/C7yjc0`,.
[15] Wireshark. `http://www.wireshark.org/`.
[16] Methodology for the subjective assessment of the quality of television pictures. ITU-R Rec. BT.500-11 , 2002.
[17] H.264: Advanced video coding for generic audiovisual service. International Telecommunication Union, 2013.
[18] White paper: Live and on-demand streaming video with lifesize uvc video center. March, 2012.
[19] Xuan Bao, Trevor Narayan, Ardalan Amiri Sani, Wolfgang Richter, Romit Roy Choudhury, Lin Zhong, and Mahadev Satyanarayanan. The case for context-aware compression. In *Proc. of HotMobile 2011*.
[20] Floraine Berthouzoz, Wilmot Li, and Maneesh Agrawala. Tools for placing cuts and transitions in interview video. *ACM Trans. Graph.*, 31(4):67:1–67:8, July 2012.
[21] Thomas Brox, AndrÃl's Bruhn, Nils Papenberg, and Joachim Weickert. High accuracy optical flow estimation based on a theory for warping. In *Proc. of ECCV 2004*.
[22] An (Jack) Chan, Amit Pande, Eilwoo Baik, and Prasant Mohapatra. Temporal quality assessment for mobile videos. In *Proc. of Mobicom '12*.
[23] Jiasi Chen, Amitabha Ghosh, Josphat Magutt, and Mung Chiang. Qava: quota aware video adaptation. In *Proc. of CoNEXT 2012*.
[24] Jiasi Chen, Rajesh Mahindra, Mohammad Amir Khojastepour, Sampath Rangarajan, and Mung Chiang. A scheduling framework for adaptive video delivery over cellular networks. In *Proc. of MobiCom 2013*.
[25] Xiaoming Chen, Zhendong Zhao, Ahmad Rahmati, Ye Wang, and Lin Zhong. Save: sensor-assisted motion estimation for efficient h.264/avc video encoding. In *Proc. of ACMMM 2009*.
[26] Hossein Falaki, Ratul Mahajan, and et al. Diversity in smartphone usage. In *Proc. of MobiSys 2010*.
[27] Shih fu Chang, Shih fu Chang, Anthony Vetro, Anthony Vetro, and Senior Member. Video adaptation: Concepts, technologies, and open issues. In *Proc. of the IEEE*, 2005.
[28] Sangtae Ha, Soumya Sen, Carlee Joe-Wong, Youngbin Im, and Mung Chiang. Tube: time-dependent pricing for mobile data. In *Proc. of SIGCOMM 2012*.
[29] Guangming Hong, Ahmad Rahmati, Ye Wang, and Lin Zhong. Sensecoding: Accelerometer-assisted motion estimation for efficient video encoding. In *Proc. of ACMMM 2008*.
[30] Mohammad Ashraful Hoque, Matti Siekkinen, and Jukka K. Nurminen. Using crowd-sourced viewing statistics to save energy in wireless video streaming. In *Proc. of MobiCom 2013*.
[31] Lorenzo Keller, Anh Le, Blerim Cici, Hulya Seferoglu, Christina Fragouli, and Athina Markopoulou. Microcast: cooperative video streaming on smartphones. In *Proc. of MobiSys 2012*.
[32] Ira Kemelmacher-Shlizerman, Eli Shechtman, Rahul Garg, and Steven M. Seitz. Exploring photobios. In *Proc. of ACM SIGGRAPH 2011*.
[33] Sunhun Lee and Kwangsue Chung. Combining the rate adaptation and quality adaptation schemes for wireless video streaming. *J. Vis. Comun. Image Represent.*, 19(8):508–519, December.
[34] Xin Li, Mian Dong, Zhan Ma, and Felix C.A.

Fernandes. Greentube: power optimization for mobile videostreaming via dynamic cache management. In *Proc. of ACMMM 2012*.

[35] Ying Li, Zhu Li, Mung Chiang, and A. Robert Calderbank. Content-aware distortion-fair video streaming in congested networks. *IEEE Trans. Multi.*, 11(6):1182–1193, October.

[36] Yao Liu, Lei Guo, Fei Li, and Songqing Chen. An empirical evaluation of battery power consumption for streaming data transmission to mobile devices. In *Proc. of ACMMM 2011*.

[37] Jia Meng. Compressive sensing in wireless communications. In *Ph.D. Thesis, UNIVERSITY OF HOUSTON, 2010, 121 pages*.

[38] Shivajit Mohapatra, Radu Cornea, Nikil Dutt, Alex Nicolau, and Nalini Venkatasubramanian. Integrated power management for video streaming to mobile handheld devices. In *Proc. of ACMMM 2003*.

[39] Yen-Fu Ou, Tao Liu, Zhi Zhao, Zhan Ma, and Yao Wang. Modeling the impact of frame rate on perceptual quality of video. In *Proc. of ICIP 2008*.

[40] Andrew J. Pyles, Xin Qi, Gang Zhou, Matthew Keally, and Xue Liu. Sapsm: Smart adaptive 802.11 psm for smartphones. In *Proc. of UbiComp 2012*.

[41] Xin Qi, Qing Yang, David T. Nguyen, and Gang Zhou. Context-aware frame rate adaption for video chat on smartphones. In *Adjunct Proc. of UbiComp 2013*.

[42] Ahmad Rahmati and Lin Zhong. Context-for-wireless: Context-sensitive energy-efficient wireless data transfer. In *Proc. of MobiSys 2007*.

[43] Lars Lau Rakêt, Lars Roholm, Andrés Bruhn, and Joachim Weickert. Motion compensated frame interpolation with a symmetric optical flow constraint. In *Proc. of ISVC 2012*.

[44] Reza Rejaie, Mark Handley, and Deborah Estrin. Quality adaptation for congestion controlled video playback over the internet. In *Proc. of SIGCOMM 1999*.

[45] Shu Shi, Cheng-Hsin Hsu, Klara Nahrstedt, and Roy Campbell. Using graphics rendering contexts to enhance the real-time video coding for mobile cloud gaming. In *Proc. of ACMMM 2011*.

[46] Wei Song, Dian Tjondronegoro, and Michael Docherty. Saving bitrate vs. pleasing users: where is the break-even point in mobile video quality? In *Proc. of ACMMM 2011*.

[47] Jonathan Su and Russell M. Mersereau. Motion-compensated interpolation of untransmitted frames in compressed video. In *Proc. of 301h Asilomar Conf. on Signals Systems and Computers, Asiloma; USA*, 1996.

[48] Yao Wang, Ya-quin Zhang, and Joern Ostermann. *Video Processing and Communications*. 2001.

[49] Li Zhang, Dhruv Gupta, and Prasant Mohapatra. How expensive are free smartphone apps? *SIGMOBILE Mob. Comput. Commun. Rev.*, 16(3):21–32, December 2012.

[50] Lide Zhang, Birjodh Tiwana, Zhiyun Qian, and et al. Accurate online power estimation and automatic battery behavior based power model generation for smartphones. In *Proc. of CODES/ISSS 2010*.