A Context-Aware Framework for Reducing Bandwidth Usage of Mobile Video Chats

Xin Qi, Qing Yang, David T. Nguyen, Ge Peng, *Student Member, IEEE*, Gang Zhou, *Senior Member, IEEE*, Bo Dai, Daqing Zhang, *Member, IEEE*, and Yantao Li, *Member, IEEE*

***Abstract*—Mobile video chat apps offer users an approachable way to communicate with others. As high-speed 4G networks are being deployed worldwide, the number of mobile video chat app users increases. However, video chatting on mobile devices brings users financial concerns, since streaming video demands high bandwidth and can use up a large amount of data in dozens of minutes. Lowering the bandwidth usage of mobile video chats is challenging since video quality may be compromised. In this paper, we attempt to tame this challenge. Technically, we propose a context-aware frame rate adaption framework, named low-bandwidth video chat (LBVC). It follows a sender-receiver cooperative principle that smartly handles the tradeoff between lowering bandwidth usage and maintaining video quality. We implement LBVC by modifying an open-source app–Linphone– and evaluate it with both objective experiments and subjective studies.**

***Index Terms*—Bandwidth reduction, context awareness, frame interpolation, frame rate adaption, mobile video chats.**

I. Introduction

MOBILE video chat apps provide users with approachable ways to communicate with others. It has been presented that mobile video chat apps are among the most popular smartphone apps [10]. Meanwhile, the number of mobile video chat users continuously grows as mobile devices, such as smartphones, become prevalent [6], [7]. As 4G high-speed network services are deployed worldwide, network speed is no longer a major bottleneck of performing mobile video chats. Companies, such as Snapchat, that have mobile video chat apps as products draw large attractions from investigators because of their large user populations and potential commercial values.

However, "*mobile video chat apps consume much more bandwidth than other apps*" [11]. Take Skype as an example, the upload and download bandwidth requirements for its low quality video chats are both 300 Kbps [8]. The bandwidth usage at this level can quickly use up any existing limited data plan within couple of hours. As alleviations, video encoding techniques [12]–[15] have been proposed for reducing video streaming bandwidth. Other works [9], [16] propose solutions that further lower the bandwidth usage of video downloading based on existing encoding techniques. Our previous work [46] indicates that bandwidth usage of mobile video chats can be further reduced by dynamically adapting video frame rate based on smartphone vibration.

In this paper, we extend our previous design by proposing a general context-aware frame rate adaption framework, named LBVC (Low-bandwidth Video Chat). It introduces guarding context as a replaceable module that could be set as whatever context developers feel proper for video frame rate adaption. With guarding context configured, it lets the video sender adapt frame rate with respect to user-selected guarding context and frame rate to lower bandwidth usage and video receiver interpolate 'missing' frames to maintain video quality. LBVC also provides a user-friendly interface to help users select a guarding context to control frame rate and set an appropriate frame rate to save bandwidth. 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 context-aware sender-receiver cooperative approach. The sender utilizes the lower user input frame rate to encode video chats and save bandwidth unless the user selected guarding context is detected. Once the guarding context is detected, it switches to the higher default frame rate. In this way, LBVC aggressively reduces bandwidth usage at video sender and does not provide the accurate quality guarantee of outgoing videos. The receiver makes up the missing information in the received video chats by interpolating intermediate frames when the instant frame rate is smaller than the default one. According to existing literature [19], frame interpolation technique adopted by receiver may produce strong artifacts when the scene change between consecutive frames is large. However, the frame rate adaption at the sender is designed to keep the scene change between consecutive frames small in order to prevent strong artifacts from the frame interpolation.

Manuscript received December 31, 2015; revised May 03, 2016; accepted May 19, 2016. Date of publication May 24, 2016; date of current version July 15, 2016. This work was supported in part by the U.S. National Science Foundation under Grant CNS-1253506 (CAREER) and Grant CNS-1250180, in part by the National Natural Science Foundation of China under Grant 61572048 and Grant 61402380, and in part by the Fundamental Research Funds for the Central Universities under Grant XDJK2013C116. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Liang Zhu. *(Corresponding author: Daqing Zhang.)*

X. Qi is with the NSX Group, VMware, Inc., Palo Alto, CA 94304 USA (e-mail: qix@vmware.com).

Q. Yang, D. T. Nguyen, G. Peng, and G. Zhou are with the Department of Computer Science, The College of William and Mary, Williamsburg, VA 23187 USA (e-mail: xqi@cs.wm.edu; qyang@cs.wm.edu; dnguyen@cs.wm.edu; gpeng@cs.wm.edu; gzhou@cs.wm.edu).

B. Dai is with the School of Computational Science and Engineering, College of Computing, Georgia Institute of Technology, Atlanta, GA 30332 USA (e-mail: bodai@gatech.edu).

D. Zhang is with the Key Laboratory of High Confidence Software Technologies, Peking University, Beijing 100871, China (e-mail: dqzhang@sei.pku.edu.cn).

Y. Li is with the College of Computer and Information Sciences, Southwest University, Chongqing 400715, China (e-mail: yantaoli@foxmail.com).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TMM.2016.2572001

The main contributions of this paper are summarized as follows.

1) We introduce guarding context for frame rate adaption in mobile video chat apps. Based on guarding context, we propose LBVC, a general context-aware frame rate adaption framework, to reduce bandwidth usage of mobile video chats.
2) We implement LBVC by modifying an existing open-source mobile video chat app, Linphone. The experiment results demonstrate that LBVC saves bandwidth by 35% compared to the default solution without obviously compromising video quality.

The rest of the paper is organized as follows. We first present the design of the LBVC framework. Next, we present its implementation and evaluation. We finally elaborate existing works and conclude the paper.

II. LBVC Design

Based on the measurement results in [46], lowering frame rate efficiently reduces bandwidth usage below the typical frame rate range (12–20 fps) adopted by existing mobile video chat apps. Video compression techniques take limited effects on compressing video chats in this frame rate range. The LBVC framework is designed to exploit the opportunity for reducing bandwidth usage by adapting frame rate within this low frame rate range as well as not obviously compromising video quality.

A. Guarding Context

In LBVC, the video quality at the receiver is determined by how well the frame interpolation algorithm works. Usually, the algorithm works well when the scene changes between consecutive frames are small. A guarding context in LBVC should be carefully designed by app developers to indicate whether the scene changes between the consecutive frames are large or not. The scene changes are large only when the guarding context is detected.

In LBVC, we give app developers the freedom to design guarding contexts. However, we particularly consider two guarding contexts in this paper, quick object movement and severe device vibration. A guarding context may work well in one usage scenario but may not in another. In the rest of this section, we will introduce how LBVC detects each guarding context and explain the reasons of why we define them through stating their working and non-working usage scenarios.

Quick object movement: LBVC detects the quick object movement by computing the metric based on the frame difference within a time period [see (1)]. In the equation, $||*||_F$ denotes the Frobenius Norm of a matrix and $start$ and end denote the starting and ending time of each measuring period. Each frame I is of size $M \times N$ pixels. An object movement is considered as quick if the metric value of a time period is beyond a threshold. Otherwise, it is considered to be not quick. This metric measures the overall effects of scene changes during a time period and approximately reflects whether objects within a scene move quickly or not. Although more complicated object detection algorithms exist, this simple approximate algorithm is chosen since it is fast and able to meet the real-time requirement of video streaming

$$A = \sum_{t=start+1}^{end} \left(\frac{||I_t - I_{t-1}||_F}{\sqrt{M * N}} \right). \tag{1}$$

The reason we define this guarding context is that it works well when the object in a video frequently moves but the smartphone is relative stationary. However, it requires multiple frames to detect object movement. Waiting for multiple frames adds response delay to scene changes. For example, LBVC needs to wait 260 ms for even one frame at 4 fps. Therefore, this guarding context does not work well when a user hold the device on his/her unstable hands, since the frequently scene changes are not responded in a timely manner at the sender and the frequent added delays impair perceived video quality at the receiver.

Severe device vibration: Inertial sensor readings from both accelerometer and gyroscope on smartphones are leveraged by LBVC to detects severe device vibrations. Technically, we define two metrics based on the sensor reading changes [20] to measure device vibrations [see (2) and (3)]. In the equations, L_1 norms of the three-dimension changes in both acceleration and ($\partial A_x(t)$, $\partial A_y(t)$ and $\partial A_z(t)$) and angular velocity ($\partial G_x(t)$, $\partial G_y(t)$ and $\partial G_z(t)$) are calculated. In the equation, $start$ and end denote the starting and ending time of each measuring period

$$m_A = \sum_{t=start}^{end} [|\partial A_x(t)| + |\partial A_y(t)| + |\partial A_z(t)|] \tag{2}$$

$$m_G = \sum_{t=start}^{end} [|\partial G_x(t)| + |\partial G_y(t)| + |\partial G_z(t)|]. \tag{3}$$

In LBVC, we set the sensors sampling rate as 50 Hz and the measuring period as 60 milliseconds. This measuring period is small enough to timely capture smartphone vibrations. A device vibration is considered *severe* when the value of either m_A or m_G is beyond a threshold. Otherwise, it is considered *not severe*.

We adopt L_1 norm because it captures the smartphone vibrations signaled by sensor reading changes in any measured dimension. We have also investigated the L_2 norm of the inertial sensor reading changes as the guarding context. It achieves similar experimental results as L_1 norm. We finally choose L_1 norm since its computation is simpler than L_2 norm and hence able to make a quicker decision. We also consider more complicated models such as Support Vector Machine or other classification models. However, we do not choose them for the following reasons, although they have the potential of achieving higher accuracy than L_1 norm. First, their higher accuracy depends on the training data, which users need to manually collect and label in various scenarios. Second, we already achieve satisfying results with L_1 norm, which is simple and general to all users. The performance gain of complicated models is limited. Third, complicated model needs more computation, which results in more execution time. Therefore, we choose L_1 norm, which is computationally efficient, simple and user-friendly.

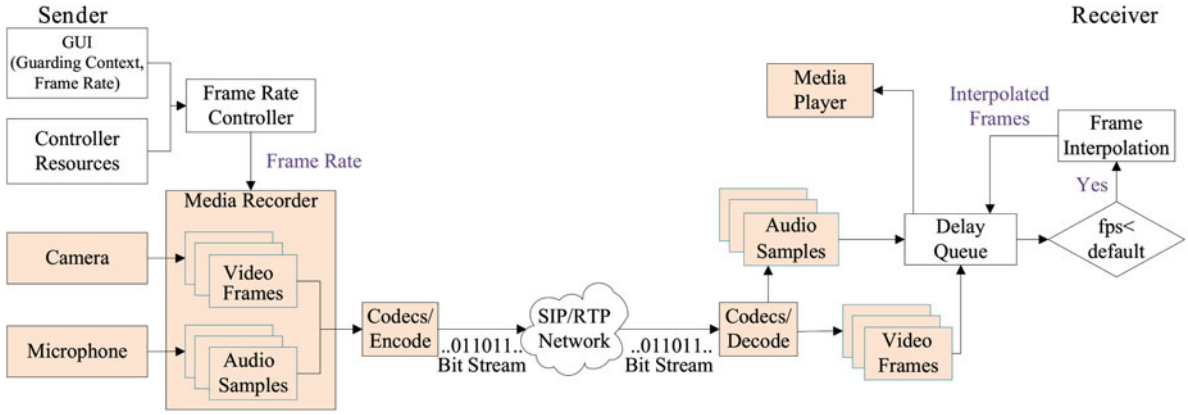


Fig. 1. LBVC architecture.

The reason we define this guarding context is that it works well when a user hold the device on his/her unstable hands, since it responses to scenes changes quickly and the delay is constantly 60 ms. However, it does not work well when the object in a video frequently moves but the device does not, since the scene changes in this case are not caused by device vibrations and hence ignored.

Finally, frame interpolation quality has different sensitivities to smartphone vibrations at different frame rates. For instance, the interpolation algorithm may produce intermediate frames with limited artifacts under typical device vibrations when the input frame rate is set as large as 8 fps. However, it may generate intermediate frames with strong artifacts under typical device vibrations when the input frame rate is set as small as 1 fps. Therefore, app developers should carefully set the threshold values for each guarding context at different frame rates. We suggest the threshold values are determined experimentally with the objective of preventing obvious artifacts.

B. LBVC Architecture

Fig. 1 depicts the architecture of LBVC with the common video chat app components in red and newly introduced components described in white rectangles and triangles. During a video chat, the communication is bi-direction and a device is both a sender and receiver. Before going to design details, we claim that the LBVS design is generic for typical video chat apps on mobile devices, and is independent of lower layer components such as codecs and network protocols.

At sender side: Users can select a guarding context and set the frame rate for a video chat through a user-friendly interface. The interface describes what each guarding context is. To be comprehensive, it also provides the information for the scenarios under which it works well and the ones under which it does not. A user selected frame rate is usually lower than the default frame rate. A lower candidate frame rate reduces bandwidth usage at the expense of degrading the video quality on conversation partner side. Users not only need to know how much bandwidth they can save by selecting a frame rate, but also they should be aware of how much damage the lower frame rate will bring to video quality. Therefore, we design the interface to provide the estimated bandwidth usage and associated video quality degradation of each candidate frame rate. The baseline is the bandwidth usage and video quality at the default frame rate.

A Frame Rate Controller controls the frame rate of the media recorder switching between the higher default frame rate and lower user input frame rate with respect to a user-selected guarding context. It utilizes the user input frame rate as the target frame rate and continuously detects whether the selected guarding context happens or not in run-time. Different guarding context detections require different resources. For example, detecting guarding context of severe device vibration requires collecting inertial sensor readings, while detecting guarding context of quick object movement requires logging recent frame difference. As we introduce in Section II-A, the scene changes are determined to be larger when the guarding context is detected. The Controller 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 the guarding context is detected.

At receiver side: Video frames and audio samples are decoded from the data packets received over the SIP/RTP network. To rescue video quality, we design the receiver to interpolate intermediate frames between the received frames whenever the received frame rate is lower than default frame rate. The frame rate after frame interpolation is not smaller than the default frame rate.

The frame interpolation algorithm at receiver side plays a key role in LBVC. At the beginning, we investigate two optical-flow [19], [21] based algorithms. However, their complicated assumptions and algorithm designs result in long execution time, which is not practical for real-time video chats. For example, the algorithm in [21] takes 825 seconds on average to interpolate one frame on a laptop, whose hardware configuration are Intel Core i7-2760QM CPU @ 2.4GHz $\times 8$ and a memory of 12 Gigabytes. The input frame resolution is similar to that in smartphone video chats. This configuration is more advanced than state-of-the-art smartphones such as Nexus 5, but optical-flow based algorithms still take such a long execution time on it. More execution time results of these algorithms are listed in [22].

After the aforementioned investigation, we turn to the cross dissolve algorithm [23]. The cross dissolve algorithm is simple

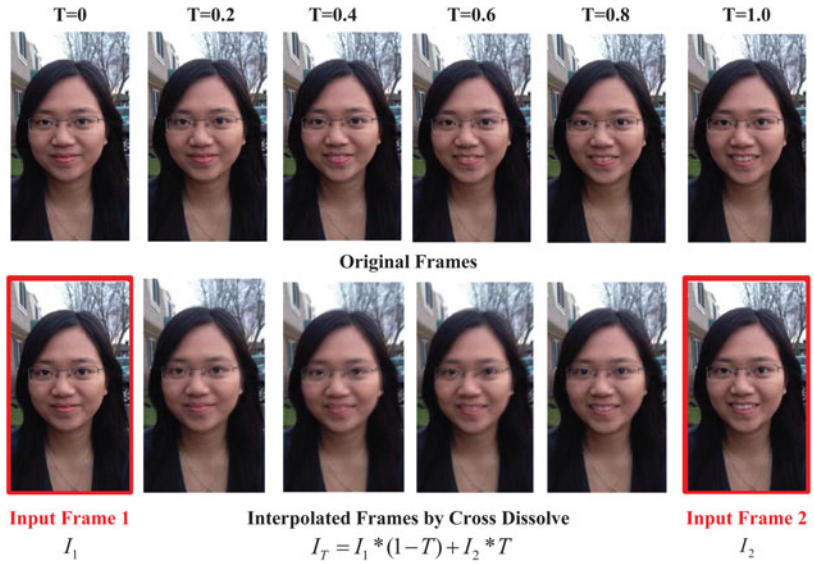


Fig. 2. Cross dissolve example. The time interval between the two input frames is 300 ms.

and shown in (4). In the equation, there are two input frames, I_1 and I_2. The interpolated frame is the weighted average of the two input frames, where the weights depend on the time point which the interpolated frame should locate at. From the equation, the algorithm only involves the operations of adding two frames and multiplying frames and constants

$$I_{\text{interpolated}} = (1-t) * I_1 + t * I_2. \tag{4}$$

Theoretically, we investigate when this algorithm is able to generate smooth object motion among the input frames and interpolated frames. Particularly, we focus on analyzing when the algorithm produces smooth object edge motion, which is a dominate factor of smooth object motion. According to existing literature [23], real image edge tends to be smooth during image process as a result of the blurring effects [32]. Mathematically, a blurring edge could be represented by a sine curve and the cross dissolve algorithm is converted into (5) [23]

$$\zeta \sin(\alpha x + \kappa) = (1-t)\eta \sin(\alpha x) + t \sin(\alpha x + \theta). \tag{5}$$

On the right hand side of this equation, there are two edges represented by two sine curves. The left edge ($\eta \sin(\alpha x)$) is the first input edge and the right edge ($\sin(\alpha x + \theta)$) is the second input edge. η is the amplitude scale and θ is the spatial translation (or phase shift). The interpolated edge is $\zeta \sin(\alpha x + \kappa)$, which represents a sequence of sine curves. The parameter details of the interpolated edge are listed in (6) and (7) [23]

$$\kappa = \arctan \frac{t \sin\theta}{(1-t)\eta + t\cos\theta} \tag{6}$$

$$\zeta^2 = \eta^2(1-t)^2 + t^2 + 2(1-t)\eta t \cos\theta. \tag{7}$$

The parameter κ controls the motion speed of interpolated edge and it is an approximately increasing function of θ [23], which is the spatial transition between the two input edges. Thus, small κ (or small θ) results in smooth transition between two input edges. The parameter ζ is the image contrast and it is an decreasing function of θ. Small ζ (or larger θ) results in unclear edge, which is termed as ghosting effect [42]. To sum up, the smaller the spatial transition θ is, the smoother and clearer the interpolated frames are. Therefore, we design the Frame Rate Controller on the sender to prevent large scene changes between consecutive frames. In Fig. 2, we provide an intuitive example of the input and output of the cross dissolve algorithm.

Another thing to be considered is the delay introduced by frame interpolation. Usually a frame interpolation technique requires two frames as input. When one frame is received, it cannot start interpolation until the next frame arrives. At the media player, in order to ensure a constant video playing speed, the first frame cannot be played until the arrival of the next frame and the accomplishment of the frame interpolation. Thus, we add a delay to the media player at the beginning of each video chat to accommodate the delay introduced by frame interpolation. For synchronization purpose, we add the same delay to audio playing. The length of the common delay is determined based on the sender-side input frame rate. For example, the receiver obtains the selected frame rate (4ps) from the sender through SIP negotiation parameters [18] at the beginning of each video chat and configure the delay (250 ms) with tolerant considerations (10 ms) about dynamic networking. Overall, a 260 ms (250 ms+10 ms) delay is added by receiver when 4 fps is selected by users.

III. LBVC Implementation and Evaluation

A. Implementation and Configuration

We have investigated several open-source mobile video chat apps, including Linphone, SipDroid and CSipSimple. We choose Linphone[1] 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 sender

[1]"Linphone," [Online]. Available: http://www.linphone.org

side interface with Android SDK 4.2 in Java and integrate it to the Android version of Linphone. The modified Linphone is installed on a Galaxy Nexus and Nexus 4 for experiments.

Linphone adopts the *Mediastreamer2* library for video streaming. In the native *Mediastreamer2* library, the video streaming procedure is implemented as a sequence of filter graphs. Each filter in a graph is in charge of one task such as video capturing, encoding, or displaying. At the beginning of each video stream, a thread called *Ticker* is scheduled to wake up every 10 ms and executes the filter graphs. We add the Frame Rate Controller as a new filter at the beginning of the Ticker thread and the Frame Interpolation Component as another new filter between the video decoding and displaying filters.

The Frame Rate Controller filter collects the necessary resources for guarding context detection. For the guarding context of *severe device vibration*, it collects accelerometer and gyroscope readings through the Android NDK sensor library API. It determines whether the calculated vibration metrics are larger than the specific thresholds or not every 60 ms. The severe device vibration is detected when the metrics are larger than the thresholds. For the guarding context of *quick object movement*, it logs the most recent frame and computes its difference from the next coming frame. Within each frame interval, it determines whether the frame difference is beyond a threshold or not. The quick object movement is detected when the frame difference is beyond the threshold. For both guarding contexts, if they are detected, the default frame rate (12 fps) is set for the video capturing and encoding filters; otherwise, the user-specified lower frame rate is set. In this section, we evaluate LBVC with severe device vibration selected as the guarding context through a series of experiments.

Table I summarizes 5 typical frame rates that are lower than the default 12 fps. For each selected frame rate, the thresholds for the Frame Rate Controller to make frame rate adaption decisions are listed. Also, the common delays added for each frame rate to synchronize the audio and video are also added listed. We suggest app developers to determine these values empirically with considerations about bandwidth saving, video quality degradation and networking dynamics.

In experiment section, we compare various results when LBVC is enabled and disabled at each selected frame rate. When LBVC is disabled, Linphone uses H.264 to encode video chats by default. H.264 is a start-of-the-art encoding method that adopts motion compensation to compress video size [13]. From the differences between the results when LBVC is enabled and those when LBVC is disabled, we are able to quantitatively infer how much cost LBVC pays for maintaining video quality.

B. Performance Under Typical Scenarios

In this section, we evaluate the performance impact of LBVC at the selected frame rates listed in Table I. Technically, we recruit the same 10 pairs of subjects and each pair performs 6-minute video chats at all selected frame rates under typical video chat scenarios such as sitting and standing for bandwidth and power measurements.

TABLE I
SELECTED FRAME RATES AND THEIR ASSOCIATED PARAMETERS (THRESHOLDS AND DELAY)

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

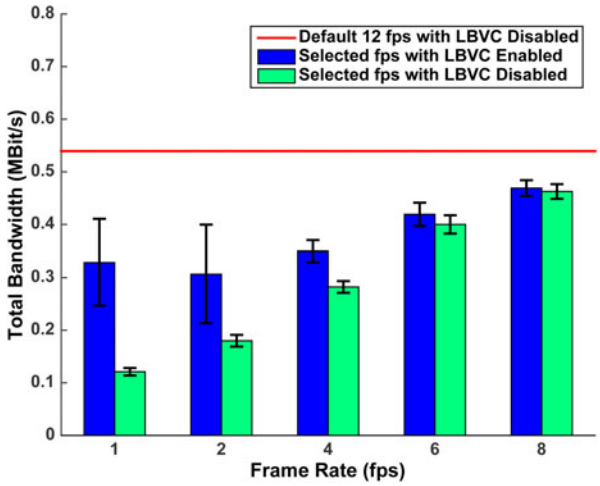


Fig. 3. Bandwidth usage versus frame rate on Galaxy Nexus.

We perform bandwidth usage and power consumption measurements over the T-Mobile HSPA+ cellular network, but only perform power consumption measurements over a public college WiFi network. WiFi network is not considered for bandwidth usage measurement since there is no bandwidth cost for video streaming under WiFi. During all video chats, we disable all unnecessary apps, services, and radios on smartphones.

1) Bandwidth Usage: To measure bandwidth usage, we adopt *Shark for Root*[2] to record network traffic and WireShark[3] to analyze bandwidth usage on a server. We separate the video chats for bandwidth measurements from those for power measurements, since the network traffic capturing software also consumes power.

Fig. 3 illustrates the measured total bandwidth usage of making video chats on Galaxy Nexus. The green bars illustrate the average total bandwidth usage at the selected frame rates with LBVC disabled, while blue bars illustrate that with LBVC enabled. The red line depicts the average total bandwidth usage at the default 12 fps with LBVC disabled.

In the figure, we observe that the bandwidth usage with LBVC enabled is higher than that with LBVC disabled. The extra bandwidth usage is actually the cost LBVC pays to maintain video quality. It is because that LBVC dynamically adapts frame rate between the lower user input frame rate and higher default frame rate. Compared to the bandwidth usage at the default frame rate, LBVC still significantly saves bandwidth. For example, the average bandwidth usage of video chats is reduced by 35% at 4 fps under typical video chat scenarios. We summarize the average bandwidth savings in Table II. The average bandwidth

[2]"Shark for root," [Online]. Available: http://goo.gl/IRCUVw
[3]"Wireshark," [Online]. Available: http://www.wireshark.org/

TABLE II
AVERAGE BANDWIDTH SAVINGS VERSUS FRAME RATES ON GALAXY NEXUS AND NEXUS 4 (COMPARED TO THE DEFAULT 12 FPS WITH LBVC DISABLED)

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

savings for the two smartphones are similar because the communication is symmetric. Additionally, lower bandwidth usage at clients reduces the traffic flow competition at the RTP server, and potentially results in better networking conditions.

We also 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 device 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 device 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.

2) Power Consumption: In the experiments, we utilize the Monsoon Power Monitor[4] 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 [24] 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.

Figs. 4 and 5 summarize the measured average power consumption of all the video chats over the cellular and WiFi networks on Galaxy Nexus. The green bars illustrate the average power consumption at the different frame rates with LBVC enabled, while the blue bars depict that with LBVC disabled. In the figures, the power consumption at 12 fps is plotted as the base line, which is to be compared with the power consumption at other selected frame rates.

From the figures, as we expect, the power consumption increases as frame rate increases. At each frame rate, the power consumption when LBVC is enabled is higher than that when LBVC is disabled, since the sensing and frame interpolation operations in LBVC consumes extra power. We also 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. Also, due to the unstable runtime frame rate, the power consumption variances with LBVC enabled at 1 fps is larger than those at higher frame rates.

[4]"Monsoon power monitor," [Online]. Available: http://goo.gl/spM1pt

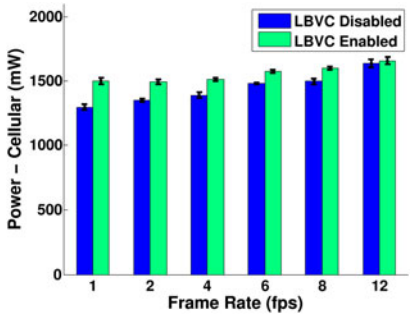


Fig. 4. Power consumption versus frame rate in a cellular network on Galaxy Nexus.

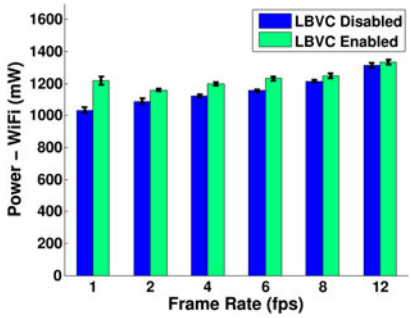


Fig. 5. Power consumption versus frame rate in a WiFi network on Galaxy Nexus.

TABLE III
AVERAGE POWER SAVINGS VERSUS FRAME RATES ON GALAXY NEXUS AND NEXUS 4 (COMPARED TO THE DEFAULT 12 FPS WITH LBVC DISABLED)

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

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

C. Video Quality

We organize a user study to investigate the impact of the LBVC on both objective and subjective video quality. In the study, we recruit 21 different pairs of subjects and there are 42 subjects in total. Each pair performs 6-minute video chats with the LBVC being either disabled or enabled. The video chats are done under different selected frame rates. To guarantee subject

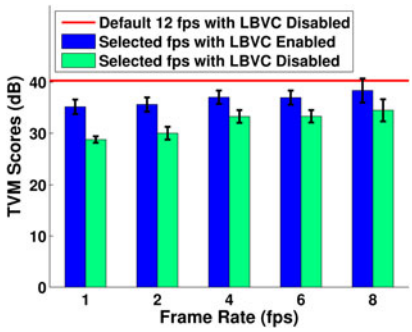


Fig. 6. TVM scores versus frame rate. Error bars indicate the standard deviations.

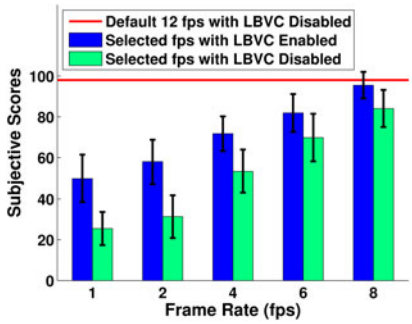


Fig. 7. Subjective scores versus frame rate. Error bars indicate the standard deviations.

diversity, we recruit half of the subjects from Physics, Applied Science, Mathematics, Nursing, Education and Rhythmic Gymnastic departments and the other half from Computer Science departments. During each video chat, we collect both objective and subjective scores as follows.

For objective scores, we implement a video chat recorder and add it to the *Mediastreamer2* library in Linphone. It automatically records all video chats during the study. We select the Temporal Variation Metric (TVM) [17] to measure the objective video quality, since it is a start-of-the-art video quality metric for video streaming on mobile devices. The recorded videos are analyzed by MATLAB on a serve to obtain the TVM scores.

To collect subjective scores, we carry out a Double Stimulus Continuous Quality Evaluation (DSCQE) [25]. Technically, each subject rates a video chat by comparing it to a reference video. The scores range is from 0 to 100 and can be explained as: very annoying (0–20); annoying (21–40); fair (41–60); good (61–80); excellent (81–100).

In DSCQE, each subject pair perform a video chat under the default 12 fps with LBVC disabled. This video chat is used as the reference baseline. Then, we randomly choose a frame rate from Table I. Each subject pair perform and score a video chat with the Linphone being configured with that frame rate when LBVC is either enabled or disabled. We repeat all these steps until all the selected frame rates have been traversed. In addition, we also collect the subjects' opinions about the delay they experience during each video chat.

We summarize the objective and subjective scores in Figs. 6 and 7. In the figures, the green bars illustrate the average scores at the selected frame rates with LBVC disabled, the blue bars

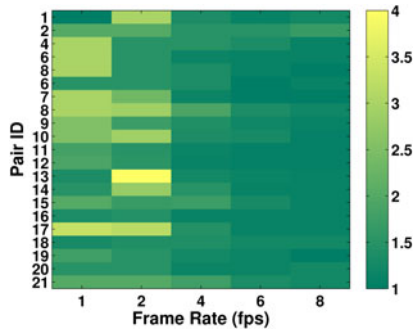


Fig. 8. Ratio of the subjective score with LBVC enabled to the subjective score with LBVC disabled for each pair.

illustrate those with LBVC enabled and the red line depicts the average score at the default 12 fps of Linphone with LBVC disabled.

Objective scores: Fig. 6 summarizes the TVM scores calculated from the recorded videos. From the figure, we observe that the TVM score at 4 fps with LBVC enabled is much higher than that with LBVC disabled. Since TVM score is defined based on the log of the difference between consecutive frames, it means LBVC can significantly reduce the difference between consecutive frames at lower frame rate. Videos become smoother as a result of the significant decrease in the mean square difference between consecutive frames. Thus, we conclude that LBVC is able to objectively alleviate video quality degradation at lower frame rate.

Subjective scores: Fig. 7 illustrates the average scores given by the subjects involved in the user study. Take 4 fps as example, LBVC increases the subjective score by 25.3% compared to that when it is disabled. The score increasing percentage is even higher at 6 fps. Thus, we conclude that LBVC is able to subjectively alleviate the perceptual video quality degradation at lower frame rate.

From the figures, we observe that the subjective score is harder for LBVC to maintain than the objective score as frame rate decreases. Here are two possible explanations. First, the lower the frame rate is, the larger the scene change between consecutive frames is. Larger scene changes result in more artifacts, which may impair the subjects' perceptual experience. Second, the threshold values are smaller at lower frame rates (e.g., 1 and 2 fps), and they result in the frequent changes between the default frame rate and input frame rate. The frequent frame rate changes may also impair the subjects' perceptual experience.

Fig. 8 is a heat map of the subjective score ratios with LBVC enabled and disabled at each frame rate. We observe that LBVC alleviates video quality degradation most of the timer. Additionally, from both Figs. 7 and 8, we observe that at lower frame rate such as 1 fps and 2 fps, the video quality improvement percentages are larger than those at other frame rates. This is because the videos are extremely jerky at 1 and 2 fps, and LBVC can accentuates the smoothness by frame rate control and frame interpolation. From Figs. 3–5 we observe that LBVC pays extra cost for maintaining video quality at each frame rate. To better understand how much bandwidth LBVC pays for every improved unit of subjective

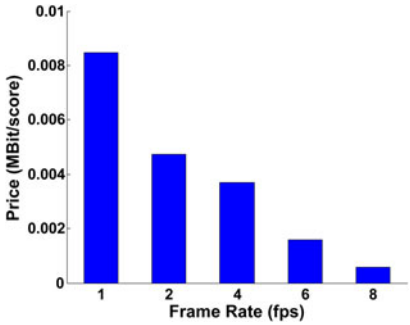


Fig. 9. Average price in terms of bandwidth (MBits) LBVC pays for every improved unit of subjective score.

score, we present Fig. 9. This figure demonstrates the average price, in terms of bandwidth (MBits), LBVC pays for every improved unit of subjective score. From the figure, we observe that as the frame rate decreases, the price increases. It is because that at lower frame rates, effective video content is limited and video quality is far from being acceptable. Thus, LBVC needs to spend more bandwidth to collected additional video content for improving subjective score by one unit. In contrast, at higher frame rates, there is more video content. With small amount of additional bandwidth, LBVC is able to gain an extra unit of subjective score.

Finally, from the users' feedbacks, the common delays added at the beginning of each video chat are acceptable. They are even negligible when the selected frame rate is 4 fps and above.

From Tables II and III, and Fig. 7, 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.*

IV. Related Work

The original conference version [46] of this manuscript proposes a vibration-aware frame rate adaption framework for achieving low-bandwidth mobile video chats. The conference paper is not comprehensive enough since its effectiveness fails when users make video chat with smartphone in stationary status, such as being placed on a table. Therefore, we extend the conference paper to a general context-aware frame rate adaption framework by introducing the concept of guarding contexts. General guarding contexts are used for keeping scene changes small and controlling video frame rate. In addition to the guarding context of device vibration appeared in the conference paper, we add the guarding context of quick object movement into the system. This new guarding context is used for dealing with the situation where smartphone is in stationary status. Additionally, we present the theoretical analysis for the effectiveness of the frame interpolation algorithm we adopt. Finally, we make more efforts on the experiments and update the figures in the experiment section in order to incorporate the new results.

In this section, we discuss how LBVC is different from other existing literatures from the perspectives of video bitrate adaption, video streaming bandwidth reduction, video streaming power reduction and image interpolation.

Video bitrate adaption: Existing video bitrate adaption techniques optimize video bandwidth usage by considering various contextual information, such as networking conditions [1][2], [5], [30], [31], video contents [33], user preferences [4], [38], and residual power [3]. Being aware of these contexts, they control video streaming bitrate in order to maximize video utility (or video quality) for video consumers [39]. 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.

Recent works, such as QAVA [9] and TUBE [40] help user economically use their data plan quota for video streaming. LBVC is different from them from the following two perspectives. First, LBVC is designed to optimize the bandwidth usage of streaming *live* and *interactive* video chats, but QAVA and TUBE are designed for optimizing videos *downloading*. Second, QAVA and TUBE optimize bandwidth usage as well as guarantee the quality of outgoing videos by making trade-off between video quality and bitrate (or bandwidth usage) at video sender. Instead, LBVC takes a more aggressive frame rate adaption approach to reduce video bitrate at video sender. It does not guarantee the quality of outgoing videos, but it adopts extra techniques at both video sender (context-aware frame rate control) and receiver (frame interpolation) to maintain the video quality.

Video streaming bandwidth reduction: Video compression techniques [12], [13], [26], [27], [14] have also been proposed to reduce video streaming bandwidth usage. Some of them, such as VP8 [12] and H.264 [13], reduce video sizes by leveraging motion compensation technique that exploits the temporal correlation among frames. Furthermore, SenseCoding [27] and SaVE [26] utilize inertial sensors, such as accelerometers, to enhance the aforementioned motion estimation based methods. Additionally, techniques such as compress sensing [28] and context-aware compression [29] have also been proposed for compressing videos.

The main goal of video compression techniques is to reduce the number of necessary bits to encode videos by exploiting various contextual information from video frames. Instead, LBVC takes a different approach by reducing the input size of video compression techniques and this approach is orthogonal to the existing video compression techniques.

Video streaming power reduction: Empirical measurements performed in [41] 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. In [34], the authors propose an approach that jointly utilizes unicast and multicast to optimize the energy consumption of video streaming on mobile devices. SMMC [35] optimizes mobile energy consumption by considering demand, socialization, mobility, and delivery simultaneously. It achieves high content delivery efficiency and QoS at the same time. QUVoD [36] is a QoE-driven User-centric solution for Video-on-Demand (VoD) services in urban vehicular network. It achieves high efficiency in terms of energy consumption and quality of experience. PMCV [37] achieves the good balance between energy efficiency and performance by creatively detecting and managing mobile community.

GreenTube [43] 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: The cross dissolve algorithm has been extensively used in computer graphics. For example, it has been used to interpolate intermediate frames between two well-aligned images with the purpose of creating a face animation with smooth face transitions in Exploring Photobios [23]. It has also been used in InterviewVideo [44] to provide smooth transitions between different video segmentations in an interview video editing tool. Optical-flow based warping algorithms, such as [19] and [21], are also considered in LBVC's design phase, but they are not selected because these warping algorithms usually involve complex operations, such as image derivation and matrix multiplication. It makes these algorithms computationally intensive and we finally resort to the simpler cross dissolve algorithm [23] in LBVC. Works such as [45] utilize frame interpolation techniques to interpolate the frames which are *passively lost* during video transmission. Although LBVC also interpolates frames, those frames are *deliberately dropped* to save bandwidth.

V. Conclusion

In this paper, we attempt to balance the mobile video quality and bandwidth usage by proposing LBVC, a general context-aware frame rate adaption framework. LBVC adapts the video frame rate with respect to the selected guarding context at the sender side and makes up missing information by interpolating the intermediate frames at the receiver side. We implement LBVC with guarding context of severe device vibration and demonstrate its effectiveness through real system experiments and a user study. LBVC saves bandwidth by up to 35% under typical video chat scenarios as well as alleviate video quality degradation.

References

[1] X. Wang, M. Chen, T. T. Kwon, L. T. Yang, and V. C. M. Leung, "AMES-Cloud: A framework of adaptive mobile video streaming and efficient social video sharing in the clouds," *IEEE Trans. Multimedia*, vol. 15, no. 4, pp. 811–820, Jan. 2013.

[2] H. E. Egilmez, S. Civanlar, and A. M. Tekalp, "An optimization framework for QoS-enabled adaptive video streaming over openflow networks," *IEEE Trans. Multimedia*, vol. 15, no. 3, pp. 710–715, Dec. 2012.

[3] S. Chuah, Y. Tan, and Z. Chen, "Rate and power allocation for joint coding and transmission in wireless video chat applications," *IEEE Trans. Multimedia*, vol. 17, no. 5, pp. 687–699, Mar. 2015.

[4] L. Chen, Y. Zhou, and D. Chiu, "Smart streaming for online video services," *IEEE Trans. Multimedia*, vol. 17, no. 4, pp. 485–497, Feb. 2015.

[5] A. Argyriou, D. Kosmanos, and L. Tassiulas, "Joint time-domain resource partitioning, rate allocation, and video quality adaptation in heterogeneous cellular networks," *IEEE Trans. Multimedia*, vol. 17, no. 5, pp. 736–745, Mar. 2015.

[6] L. Rainie and K. Zickuhr, "US video call statistics," Pew Research Center: Washington, DC, USA, Oct. 2010. [Online]. Available: http://goo.gl/C7yjcO

[7] "Mobile video chat user population trend in U.K.", Statista: Hamburg, Germany, 2015. [Online]. Available: http://goo.gl/YtEWR1

[8] "Skype bandwidth requirement," Skype Commun. SARL, Rives de Clausen, Luxembourg, 2016. [Online]. Available: http://goo.gl/apzY9s

[9] J. Chen, A. Ghosh, J. Magutt, and M. Chiang, "Qava: Quota aware video adaptation," in *Proc. 8th Int. Conf. Emerg. Netw. Experiments Technol.*, Dec. 2012, pp. 121–132.

[10] H. Falaki *et al.*, "Diversity in smartphone usage," in *Proc. 8th Int. Conf. Mobile Syst., Appl., Serv.*, San Francisco, CA, USA, Jun. 2010, pp. 179–194.

[11] "White paper: Live and on-demand streaming video with lifesize UVC video center," LifeSize Division, LogiTech, Mar. 2012.

[12] J. Bankoski, J. Koleszar, L. Quillio, J. Salonen, P. Wilkins, and Y. Xu, "RFC6386—VP8 data format and decoding guide," Google Inc., Mountain View, CA, USA, Nov. 2011. [Online]. Available: http://tools.ietf.org/html/rfc6386.

[13] *Advanced Video Coding for Generic Audiovisual Service,* International Telecommunication Union H.264, 2013.

[14] S. Shi, C.-H. Hsu, K. Nahrstedt, and R. Campbell, "Using graphics rendering contexts to enhance the real-time video coding for mobile cloud gaming," in *Proc. 19th ACM Int. Conf. Multimedia*, Nov./Dec. 2011, pp. 103–112.

[15] Y. Wang, Y.-Q Zhang, and J. Ostermann, *Video Processing and Communications,* Englewood Cliffs, NJ, USA: Prentice-Hall, 2001.

[16] L. Keller *et al.*, "Microcast: Cooperative video streaming on smartphones," in *Proc. 10th Int. Conf. Mobile Syst., Appl., Serv.*, Jun. 2012, pp. 57–70.

[17] A. J. Chan, A. Pande, E. Baik, and P. Mohapatra "Temporal quality assessment for mobile videos," in *Proc. 18th Int. Mobile Comput. Netw.*, Aug. 2012, pp. 221–232.

[18] M. Handley and V. Jacobson, "SDP: Session description protocol," ISI/LBNL, Apr. 1998. [Online]. Avaiable: http://goo.gl/fcphyP.

[19] T. Brox, A. Bruhn, N. Papenberg, and J. Weickert, "High accuracy optical flow estimation based on a theory for warping," in *Proc. Eur. Conf. Comput. Vision*, 2004, vol. 3024, pp. 25–36.

[20] A. Rahmati and L. Zhong, "Context-for-wireless: Context-sensitive energy-efficient wireless data transfer," in *Proc. 5th Int. Mobile Syst., Appl., Serv.*, Jun. 2007, pp. 165–178.

[21] L. L. Raket, L. Roholm, A. Bruhn, and J. Weickert, "Motion compensated frame interpolation with a symmetric optical flow constraint," in *Proc. Int. Symp. Visual Comput.*, 2012, vol. 7431, pp. 447–457.

[22] S. Baker, D. Scharstein, J. P. Lewis, S. Roth, M. Black, and R. Szeliski, "Optical-flow execution time." (2011). [Online]. Available: http://goo.gl/vcS1px.

[23] I. Kemelmacher-Shlizerman, E. Shechtman, R. Garg, and S. M. Seitz "Exploring photobios," *ACM Trans. Graph.*, vol. 30, no. 4, Jul. 2011.

[24] L. Zhang *et al.*, "Accurate online power estimation and automatic battery behavior based power model generation for smartphones," in *Proc. Int. Conf. Hardware/Softw. Codesign Syst. Synthesis*, Oct. 2010, pp. 105–114.

[25] *Methodology for the Subjective Assessment of the Quality of Television Pictures*, ITU-R Rec. BT. pp. 500–11, 2002.

[26] X. Chen, Z. Zhao, A. Rahmati, Y. Wang, and L. Zhong, "SaVE: Sensor-assisted motion estimation for efficient h.264/avc video encoding," in *Proc. 17th Int. Conf. Multimedia*, Oct. 2009, pp. 381–390.

[27] G. Hong, A. Rahmati, Y. Wang, and L. Zhong, "Sensecoding: Accelerometer-assisted motion estimation for efficient video encoding," in *Proc. 16th Int. Conf. Multimedia*, Oct. 2008, pp. 749–752.

[28] J. Meng, "Compressive sensing in wireless communications," Ph.D. dissertation, Dept. Elect. Comput. Eng., Univ. Houston, Houston, TX, USA, 2010.

[29] X. Bao *et al.*, "The case for context-aware compression," in *Proc. 12th Workshop Mobile Comput. Syst. Appl.*, 2011, pp. 77–82. 2011.

[30] J. Chen, R. Mahindra, M. A. Khojastepour, S. Rangarajan, and M. Chiang, "A scheduling framework for adaptive video delivery over cellular networks," in *Proc. 19th Int. Conf. Mobile Comput. Netw.*, Sep./Oct. 2013, pp. 389–400.

[31] S. Lee and K. Chung, "Combining the rate adaptation and quality adaptation schemes for wireless video streaming," *J. Vis. Commun. Image Represent.*, vol. 19, no. 8, pp. 508–519, Dec. 2008.

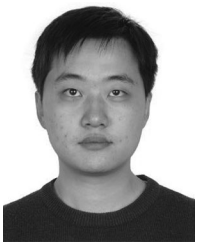
[32] D. Marr and E. HildrethI, "Theory of edge detection," in *Proc. Roy. Soc. London B*, vol. 207, pp. 187–217, 1980.

[33] Y. Li, Z. Li, M. Chiang, and A. R. Calderbank, "Content-aware distortion-fair video streaming in congested networks," *IEEE Trans. Multimedia*, vol. 11, no. 6, pp. 1182–1193, Oct. 2009.

[34] S. Almowuena, M. M. Rahman, C. H. Hsu and A. A. Hassan, "Energy-aware and bandwidth-efficient hybrid video streaming over mobile networks," *IEEE Trans. Multimedia*, vol. 18, no. 1, pp. 102–115, Jan. 2016.

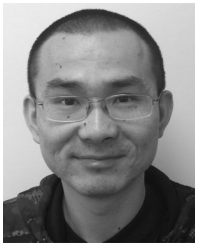
[35] C. Xu, S. Jia, L. Zhong, and G.-M. Muntean, "Socially aware mobile peer-to-peer communications for community multimedia streaming services," *IEEE Commun. Mag.*, vol. 53, no. 10, pp. 150–156, Oct. 2015.

[36] C. Xu, F. Zhao, J. Guan, H. Zhang, and G.-M. Muntean, "QoE-driven user-centric VoD services in urban multi-homed P2P-based vehicular networks," *IEEE Trans. Veh. Technol.*, vol. 62, no. 5, pp. 2273–2289, Jun. 2013.
[37] C. Xu *et al.*, "Performance-aware mobile community-based VoD streaming over vehicular Ad Hoc networks," *IEEE Trans. Veh. Technol.*, vol. 64, no. 3, pp. 1201–1217, Mar. 2015.
[38] W. Song, D. Tjondronegoro, and M. Docherty, "Saving bitrate vs. pleasing users: where is the break-even point in mobile video quality?" in *Proc. 19th Int. Conf. Multimedia*, Dec. 2011, pp. 403–412.
[39] S. F. Chang and A. Vetro, "Video adaptation: Concepts, technologies, and open issues," *Proc. IEEE*, vol. 93, no. 1, pp. 148–158, Mar. 2005.
[40] S. Ha, S. Sen, C. Joe-Wong, Y. Im, and M. Chiang, "TUBE: Time-dependent pricing for mobile data," in *Proc. Conf. Appl., Technol., Architectures, Protocols Comput. Commun.*, Aug. 2012, pp. 247–258.
[41] Y. Liu, L. Guo, F. Li, and S. Chen, "An empirical evaluation of battery power consumption for streaming data transmission to mobile devices," in *Proc. 19th Int. Conf. Multimedia*, Nov./Dec., 2011, pp. 473–482.
[42] R. Szeliski, and H.-Y. Shum, "Creating full view panoramic image mosaics and environment maps." in *Proc. Conf. Comput. Graph. Interactive Technol.*,1997, pp 251–258.
[43] X. Li, M. Dong, Z. Ma, and F. C. A. Fernandes, "GreenTube: Power optimization for mobile videostreaming via dynamic cache management," in *Proc. ACM Int. Conf. Multimedia*, Oct./Nov. 2012, pp. 279–288.
[44] F. Berthouzoz, W. Li, and M. Agrawala, "Tools for placing cuts and transitions in interview video," *ACM Trans. Graph.*, vol. 31, no. 4, pp. 671–678, Jul. 2012.
[45] J. Su and R. M. Mersereau, "Motion-compensated interpolation of untransmitted frames in compressed video," in *Proc. 30th Asilomar Conf. Signals Syst. Comput.*, Mar. 1996, pp. 100–104.
[46] X. Qi, Q. Yang, D. Nguyen, G. Zhou, and G. Peng, "LBVC: Towards low-bandwidth video chat on smartphones" in: *Proc. 6th Multimedia Syst. Conf.*, Mar. 2015, pp. 1–12.



Xin Qi received the B.Sc. degree in computer science from Nanjing University, Nanjing, China, in 2007, the M.Eng. degree in pattern recognition and intelligent systems from the National Key Laboratory of Pattern Recognition at Institute of Automation, Chinese Academy of Science, Beijing, China, in 2010, and the Ph.D. degree in computer science from the College of William and Mary, Williamsburg, VA, USA, in 2015.

He has been a Member of Technical Staff with VMware Inc., Palo Alto, CA, USA, since June 2015. His research interests include mobile systems and cloud computing.



Qing Yang received the B.S degree from the Civil Aviation University of China, Tianjin, China, in 2003, the M.S. degree from the Chinese Academy of Sciences, Beijing, China, in 2007, and is currently working toward the Ph.D degree in computer science at the College of William and Mary, Williamsburg, VA, USA.

His research interests include smartphone security and energy efficiency.

David T. Nguyen is currently working toward the Ph.D. degree in computer science at the College of William and Mary, Williamsburg, VA, USA.

Before coming to the College of William and Mary in Fall 2011, he was a Lecturer with Suffolk University, Boston, MA, USA, for 2 years. He was also a Lecturer with Christopher Newport University, Newport News, VA, USA, in 2013. In 2014, he was a Mobile Hardware Engineer with Facebook's Connectivity Laboratory, Menlo Park, CA, USA. His research interests include mobile computing, ubiquitous computing, and wireless networking.

Ge Peng (S'14) received the B.S. degree from the National University of Defense Technology, Changsha, China, in 2008, and is currently working toward the Ph.D degree in computer science at the College of William and Mary, Williamsburg, VA, USA.

Her research interests include wireless networking, smartphone energy efficiency, and mobile computing.

Gang Zhou (S'06–M'07–SM'13) received the Ph.D. degree from the University of Virginia, Charlottesville, VA, USA, in 2007.

He is an Associate Professor and Graduate Director with the Computer Science Department, College of William and Mary, Williamsburg, VA, USA. He has authored or coauthored over 70 academic papers in the areas of sensors and ubiquitous computing, mobile computing, body sensor networks, internet of things, and wireless networks. The total citations of his papers are more than 5000 according to Google Scholar, among which five of them have been transferred into patents and the MobiSys'04 paper has been cited more than 800 times.

Dr. Zhou is a Senior Member of the ACM. He is currently serving on the Journal Editorial Board of the IEEE INTERNET OF THINGS JOURNAL as well as Elsevier *Computer Networks*. He was the recipient of an award for his outstanding service to the IEEE Instrumentation and Measurement Society in 2008, the Best Paper Award of IEEE ICNP 2010, the NSF CAREER Award in 2013, and a 2015 Plumeri Award for Faculty Excellence.

Bo Dai received the B.Sc. degree from Nanjing University, Nanjing, China, the M.Sc. degree from the Institute of Automation, Chinese Academy of Sciences, Beijing, China, and is currently working toward the Ph.D. degree in computing at the Georgia Institute of Technology, Atlanta, GA, USA.

His research interests include statistical machine learning, particularly developing nonparametric statistical methods and trying to make nonparametric models scalable with provable statistical and computational guarantees.

Daqing Zhang (M'11) received the Ph.D. degree from the University of Rome "La Sapienza," Rome, Italy, in 1996.

He is a Chair Professor with the Institute of Software, Peking University, China. His research interests include context-aware computing, urban computing, mobile social networking, big data analytics, and pervasive elderly care.

Prof. Zhang served as the General or Program Chair for more than 10 international conferences, giving keynote talks at more than 16 international events. He is an Associate Editor for four journals, including ACM *Transactions on Intelligent Systems and Technology* and the IEEE TRANSACTIONS ON BIG DATA. He was the recipient of the ACM UbiComp 2015 Honorable Mention Award, IEEE UIC 2015 Best Paper Award, the Ten Years CoMoRea Impact Paper Award at IEEE PerCom 2013, the Best Paper Award at IEEE UIC 2012, and the Best Paper Runner Up Award at Mobiquitous 2011.

Yantao Li (S'11–M'12) received the Ph.D. degree in computer science from Chongqing University, Chongqing, China, in December 2012.

He is an Associate Professor with the College of Computer and Information Sciences, Southwest University, Chongqing, China. From September 2010 to September 2012, he was a Visiting Scholar supported by the China Scholarship Council, working with Professor G. Zhou at the Department of Computer Science, College of William and Mary, Williamsburg, VA, USA. His research interests include wireless communication and networking, sensor networks and ubiquitous computing, and information security.