Context-Aware and Energy-Aware Video Streaming on Smartphones

Xianda Chen, *Student Member, IEEE,* Tianxiang Tan, *Student Member, IEEE,* Guohong Cao, *Fellow, IEEE,* and Thomas La Porta, *Fellow, IEEE*

Abstract—High quality video streaming for mobile devices implies high energy consumption due to the transmitted data and the variation of wireless signals. As an example, transmissions in mobile scenarios (e.g., inside a moving bus) consumes more energy for devices than when accessing from a static environment (e.g., at home). The QoE for the user does not substantially increase when watching high bitrate videos in a vibrating environment (i.e., a moving vehicle), as the context, in this case vehicle's vibration, affects the perceived QoE. To address this problem, we propose to save energy by considering the context (environment) of video streaming. To model the impact of context, we exploit the embedded accelerometer in smartphones to record the vibration level during video streaming. Based on quality assessment experiments, we collect traces and model the impact of video bitrate and vibration level on QoE, and model the impact of video bitrate and signal strength on power consumption. Based on the QoE model and the power model, we formulate the context-aware and energy-aware video streaming problem as an optimization problem. We present an optimal algorithm which can maximize QoE and minimize energy. Since the optimal algorithm requires perfect knowledge of future tasks, we propose an online bitrate selection algorithm. To further improve the performance of the online algorithm, we propose a crowdsourcing based bitrate selection algorithm. Through real measurements and trace-driven simulations, we demonstrate that the proposed algorithms can significantly outperform existing approaches when considering both energy and QoE.

Index Terms—Video streaming, context awareness, energy saving, quality of experience

1 Introduction

NOWADAYS, video streaming has become the most popular application on smartphones. It is expected that mobile video traffic will account for over 79% of the mobile data traffic by 2022 [1]. While mobile networks offer very high peak bandwidth, video streaming over mobile networks still suffers from rapid and significant network fluctuations. To adapt for various network conditions, the Dynamic Adaptive Streaming over HTTP (DASH) protocol has been widely adopted for video streaming. With DASH, at the server side, the video is cut into a sequence of video segments, each of which is encoded with different bitrates, corresponding to different resolutions. At the client side, based on the estimated network bandwidth, the video player can dynamically determine the right bitrate for each segment, such that the video segment can be successfully downloaded before it is played to maintain good *Quality of Experience* (*QoE*).

Based on DASH, many existing bitrate adaptation algorithms [2], [3], [4], [5] focus on accurately predicting the network bandwidth, and then use higher bitrates to maintain better QoE. Although streaming video at a higher bitrate can lead to better QoE, a larger amount of data will have to be downloaded and processed on smartphones and hence, consuming more energy. Since smartphones are battery-powered, the issue of energy-efficient video streaming has received considerable attention. Energy consumption for video streaming on smartphones can be optimized from the perspective of video downloading (i.e., data communication) and video processing (i.e., video playback) [6]. Researchers have proposed various techniques to reduce the power consumption of the wireless interface during video streaming [7], [8], [9], and have proposed techniques to reduce the energy consumption of video processing on smartphones [10], [11], [12], [13].

Different from these energy-saving techniques, we propose to save energy by considering the context (environment) of video streaming; i.e., watching video on a moving bus/train or in a static environment (e.g., at home) may have different QoE requirements. On a moving vehicle where the wireless signal is weak, it costs much more energy to maintain high bitrate video streaming than at a static environment such as at home or a cafe where the wireless signal is strong. The QoE for the user may not increase too much by watching high bitrate videos in a vibrating environment such as on a moving vehicle. This is because the perception of video quality is affected by the environment such as the vibration or shaking on a moving vehicle. The vibration of the smartphone causes discomforts during video watching, which results in QoE degradation even with a high resolution. As a result, in such vibrating environments, reducing the bitrate of video streaming may significantly reduce the energy consumption, without degrading the QoE too much, and hence it is important to find a better tradeoff between QoE and energy considering the context of video streaming.

To support context-aware and energy-aware video streaming on smartphones, we have the following challenges: (1) How to model the impact of context on QoE? (2) How to select the right bitrate to minimize energy and maximize QoE? To answer the first question, we recruit

• *The authors are with School of Electrical Engineering and Computer Science, Pennsylvania State University, University Park, PA 16802. E-mail: {xuc23, txt51, gxc27, tfl12}@psu.edu.*

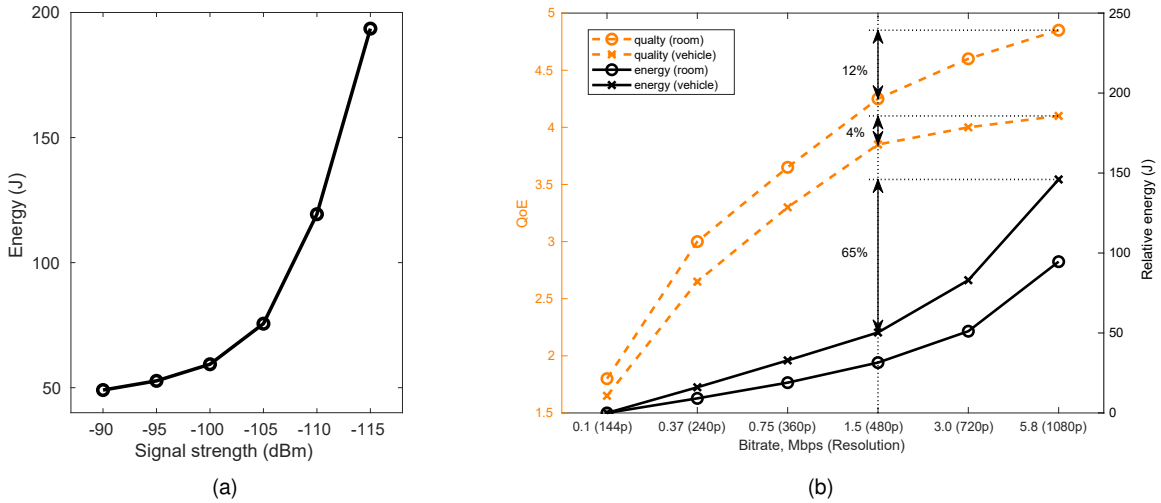


Fig. 1. (a): Total energy consumption to download 100MB data under various network conditions. (b): Perceived QoE and energy consumption as functions of bitrate under different environments (a static environment or a moving vehicle).

twenty users to watch and rate videos under two contexts; i.e., on a moving vehicle and in a static environment. To model the impact of context, we exploit the embedded sensors (e.g., accelerometer) in smartphones [14] to record the vibration level during video streaming. Based on these quality assessment experiments, we collect traces and model the impacts of video bitrate and vibration level on QoE, and model the impacts of video bitrate and signal strength on power consumption. Based on the QoE model and the power model, we formulate the context-aware and energy-aware video streaming problem as an optimization problem. We first present an optimal algorithm which can maximize QoE and minimize energy. Since the optimal algorithm requires perfect knowledge of future tasks (which will not be available in practical scenarios), we propose an online algorithm for video streaming. To further improve the performance of the online algorithm, we propose a crowdsourcing based approach for bandwidth and vibration prediction. Although crowdsourcing based approaches have been proposed for video streaming [15], [16], most of them focus on maximizing QoE, and none of them considers energy and context issues.

In summary, this paper has the following contributions.

- A study of the impact of context on both QoE and energy consumption for video streaming on smartphones. To the best of our knowledge, this is the first study of such impact.
- We formulate the context-aware and energy-aware video streaming problem as an optimization problem. We first propose an optimal solution, which provides a performance upper bound. We then propose an online bitrate selection algorithm. We also propose a crowdsourcing based bitrate selection algorithm, which predicts the available bandwidth and the vibration level using data collected from other users.
- We evaluate the proposed solutions with extensive trace-driven simulations. The evaluation results show that the proposed algorithms can significantly reduce the energy consumption while maintaining good QoE.

The rest of the paper is structured as follows. We introduce the background and motivation in Section 2. We present the system model and the problem formulation in Section 3. Section 4 presents our context-aware and energy-aware video streaming algorithm. In Section 5, we present the evaluation results. Section 6 discusses related work and Section 7 concludes the paper.

2 Background and Motivation

In DASH, the video is broken into a sequence of small HTTP-based segments, where each segment is encoded into multiple copies with various bitrates. Based on the network quality, the segment with the right bitrate is used for streaming.

Streaming video at a higher bitrate leads to better quality, but it requires to download and process more data, and thus consuming more energy. When the network condition becomes worse, much more energy will be consumed. Fig. 1(a) shows the energy consumption of downloading 100MB data under various network conditions. The measurement was done using an LG Nexus 5X smartphone using T-Mobile LTE network. In the measurement, we focus on the power consumption of the wireless interface. As can be seen, when the signal strength decreases, the energy consumption significantly increases. For example, the energy consumption increases from 49J to 193J as the signal strength changes from -90 dBm to -115 dBm.

One way to reduce the energy consumption is to use the video segment with low bitrate. However, this may reduce the video quality and affect the QoE under different contexts. To understand the impact of context on QoE and energy consumption for video streaming on smartphones, we conducted some experiments. With approval by our Institutional Review Board (IRB), twenty subjects were recruited to watch Youtube videos of various bitrates (resolutions) under two different contexts (environments):

in a static environment (room) and on a moving vehicle. After watching each video, the subjects rate the perceived quality using the nine-grade numerical quality scale (9 denotes "excellent" and 1 denotes "bad") based on the ITU-T Recommendation P.910 [17], which is then transformed to the five-level rating scale, using $MOS_5 = 1 + 4 \cdot \frac{MOS_9}{9}$. The MOS_5 value is used to represent the QoE. The energy consumption is calculated using the power models that will be detailed in Section 3.3.

As shown in Fig. 1(b), the QoE does not improve too much when the resolution is very high (e.g., 720p) for smartphones. We also see that the QoE varies with context. When the resolution drops from 1080p to 480p, the QoE degrades much slower on a moving vehicle (4%) than in a static environment (12%). On the other hand, the network condition on a moving vehicle is worse than that in a static environment. As a result, reducing the resolution from 1080p to 480p can save 65% energy when watching videos on a moving vehicle, while only degrading the QoE by 4%. Thus, it is important to consider energy, context, and QoE together for video streaming.

To model QoE considering the context of video streaming (i.e., the smoothness of the environment where users watch videos), we can exploit the embedded sensors (e.g., accelerometer) in smartphones [14]. Based on these sensors, we can differentiate whether the user is in a static environment or on a moving vehicle. We will present the detail of the QoE model and formulate the context-aware video streaming problem in the next section.

3 System Model and Problem Formulation

In this section, we introduce the video, QoE, and power models, as well as the problem formulation.

3.1 Video Model

We follow the DASH video streaming process. The video is broken into a sequence of n HTTP-based segments, where each segment contains L seconds of video. Each segment has V versions of copies corresponding to V different bitrates on the server side. Based on the network quality, the client requests the segment with the right bitrate level from the server. More specifically, the video streaming process is modeled as n data transmission tasks, corresponding to transmitting n video segments. Let T_i denote the i^{th} task, and let T_i^j denote the i^{th} task where the video segment is encoded with bitrate index j ($j \in \{1, 2, ..., V\}$).

Since many users may skip or early quit during video streaming, to save bandwidth, similar to Youtube, we use a buffer threshold (β) to limit the amount of the video data to be downloaded. β is defined in terms of seconds. Before the buffered data reaches β, i.e., the video length of downloaded but not yet viewed video in the buffer is less than β, the video player can download the next segment. When the buffered data reaches β, the player stops the downloading process. It will wait until the video length of the buffered video is less than β before downloading the next segment. Let $B_i \in [0, \beta]$ denote the amount of video data in the buffer when the client requests the i^{th} segment. To avoid stall events (or rebuffering), the i^{th} segment should be completely downloaded before B_i is drained out by the video player at the client.

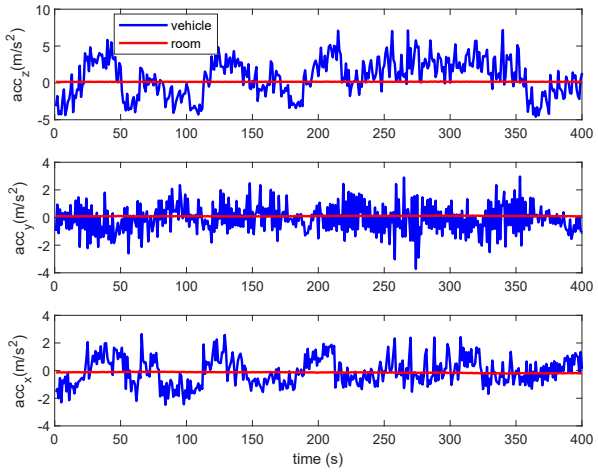


Fig. 2. Accelerometer data over time under different environments (a static environment or a moving vehicle).

3.2 QoE Model

As explained in Section 2, the QoE model should consider the context of video steaming; i.e., watching video on a moving vehicle or in a static environment may have different QoE requirements. The QoE can be modeled with Equation 1, where Q is the user perceived QoE, Q_o is the "original" quality without considering any quality loss, I_t is the quality impairment (loss) during data transmission, and I_v is the quality impairment resulting from vibration during video playback. Similar to [18], Q_o is modeled with a Michaelis-Menten function, as shown in Equation 2, where b is the bitrate, and c_1 and c_2 are the model parameters which are determined by the subjective quality assessment experiments.

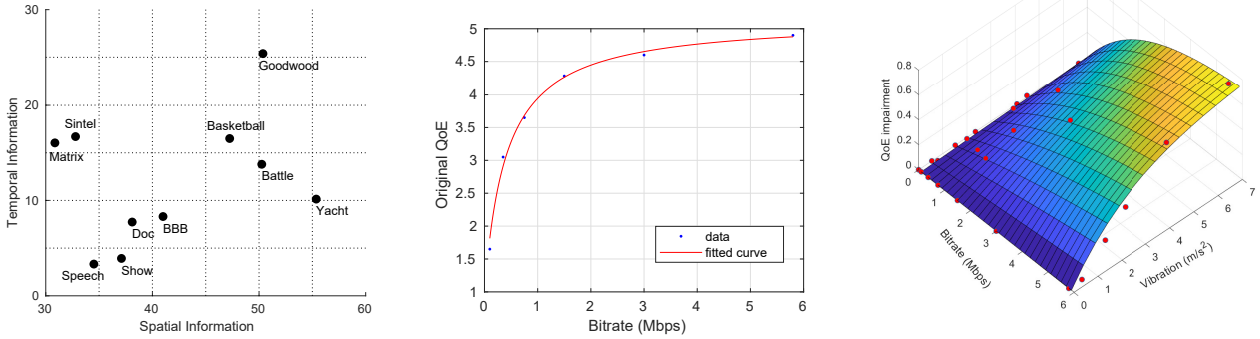
$$Q = Q_o - I_t - I_v \tag{1}$$

$$Q_o = max(1, min(5, 1 + 4 \cdot \frac{c_1 \cdot b}{c_2 + b})) \tag{2}$$

$$I_t = f_r \cdot I_r + f_b \cdot I_b \tag{3}$$

$$I_v = c_3 + c_4 \cdot exp(c_5 \cdot b \cdot v) \tag{4}$$

Similar to [19], [20], the QoE impairment during data transmission can be modeled by Equation 3, considering the impact of rebuffering events and bitrate changes. Mok *et. al* [19] has defined three levels (low, medium, and high) of rebuffering impacts on QoE by choosing I_r, and we use their base level (i.e., $I_r = 0.742$), where the rebuffering duration is less than one second. The rebuffering frequency is defined as $f_r = \frac{(S_i/R_i - B_i)_+}{B_i}$, where S_i is the segment data size, R_i is the downloading throughput, and $(x)_+ = max\{x, 0\}$. For the impact of bitrate change, we adopt the results from [20]; i.e., reducing the bitrate of a segment by 3 Mbps has the same penalty as one second rebuffering. Thus, we have



(a) Video sequences (b) Original quality (c) Vibration impairment

Fig. 3. (a): Average spatial and temporal information of the test videos. (b): The "original" quality of a video as a function of bitrate. (c): The QoE impairment due to vibration.

TABLE 1
The test videos.

Genre	Explanation	Genre	Explanation
Speech	Speech on TV	Matrix	A fight scene in The Matrix (movie)
Show	Allen show	Battle	A battle scene in The Hobbit (movie)
Doc	Documentary	Basketball	Sport
BBB	Big Buck Bunny (animation)	Yacht	Moving yacht
Sintel	Sintel (movie)	Goodwood	Horseracing

TABLE 2
The resolution and bitrate for video dataset.

Resolution	Bitrate (Mbps)
1080p	5.80
720p	3.00
480p	1.50
360p	0.75
240p	0.375
144p	0.10

TABLE 3
The parameters of QoE model.

Coefficient	c_1	c_2	c_3	c_4	c_5
Value	1.036	0.429	0.782	-0.782	0.0648

$I_b = 0.742$ and $f_b = \frac{(b_{i-1}-b_i)_+}{3.0}$, where b_i and b_{i-1} are the bitrate of the i^{th} and $(i-1)^{th}$ video segment, respectively.

To model the impact of context, we exploit the embedded sensors (e.g., accelerometer) in smartphones [14] to record the vibration level during video streaming. Since we are only interested in the acceleration associated with the vibration of the smartphone, we subtract the force caused by gravity from the raw accelerometer data. Fig. 2 shows an example of the acceleration data over time under two different environments (a static environment and a moving vehicle). As can be seen, when watching video in a static environment with the smartphone placed on the desk, the acceleration data are basically stable, while the acceleration data vary drastically due to the vibration of shaking when watching video on a moving vehicle. The vibration level is formulated with Equation 5, where m_a is the average value of M acceleration samples in the sampling window and f_a is the average variation between consecutive acceleration samples. Here we consider m_a and f_a equally; i.e., $\alpha = 0.5$. With the vibration level, we model the impact of vibration impairment with Equation 4, where b is the bitrate, v is the vibration level, and c_3, c_4 and c_5 are the model parameters.

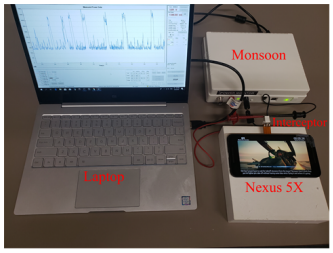
$$v = \alpha \cdot m_a + (1-\alpha) \cdot f_a \tag{5}$$

$$m_a = \frac{1}{M}\sum_{m=1}^{M}\sqrt{a_{m,x}^2 + a_{m,y}^2 + a_{m,z}^2} \tag{6}$$

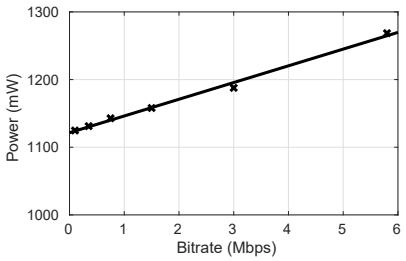
$$f_a = \frac{1}{(M-1)}\sum_{m=2}^{M}\sqrt{\sum_{l=\{x,y,z\}}(a_{m,l} - a_{m-1,l})^2} \tag{7}$$

Video Quality Assessment. To determine parameters c_1, c_2, c_3, c_4, c_5, in Equation 2 and Equation 4, we performed subjective quality assessment experiments with 20 users following ITU-T Recommendations. The subjects are asked to watch 10 videos cached in the smartphone under two contexts: in a static environment (room) and on a moving vehicle. Table 1 summaries the characteristics of the videos. To demonstrate that the chosen videos cover a wide range of different types and genres following the standard ITU-T P.910 [17], we calculate the temporal and spatial information of the videos which are shown in Fig. 3(a), where a video with higher temporal information has more changing scenes, and a video with higher spatial information has more spatial details in video frames. As can be seen, the chosen videos cover a wide range of different types and genres. The videos are encoded into different bitrate versions at 30 fps, and Table 2 shows the bitrate for each resolution. After watching a video, the subjects rate the perceived quality as discussed in Section 2. In this quality assessment experiment, the videos are locally cached in the smartphone. Then, there will not be any data transmission, and we can focus on quantifying the impact of the vibration factor.

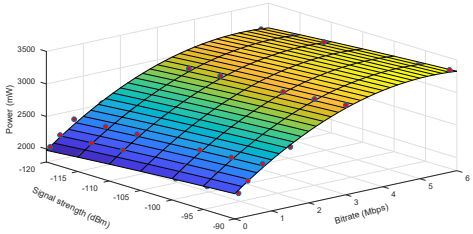
We first consider video watching in a static environment. The perceived quality of video streaming in a static environment is used as the "original" quality (i.e., Q_o), and the results are shown in Fig. 3(b). As can be seen from the figure,



(a) Setup



(b) P_b



(c) P_t

Fig. 4. (a):Experimental setup for measuring power consumption. (b): The power model when there is no data transmission. (c): The power model when there is data transmission.

Q_o increases with the increase of the video bitrate. When the bitrate becomes very high, further increasing the bitrate will not lead to significant increase in the QoE, which is consistent with the results reported in [21], [22]. With the least squares regression method, we can get the fitted curve, where the parameters c_1 and c_2 are shown in Table 3.

To see the impact of vibration on video quality, we measure the quality impairment (I_v) caused by vibration; i.e., the QoE difference between watching the same video with the same bitrate in a static environment and on a moving vehicle. Based on the accelerometer data collected by the smartphone during video watching, and Equation 5, we can calculate the vibration level (v). The relationship among QoE impairment (I_v), vibration level (v), and video bitrate (b) is shown in Fig. 3(c). As shown in the figure, when the bitrate is very small (e.g., 0.1 Mbps), the QoE is very poor regardless of the context and thus the vibration impairment is almost zero. When the vibration level is very low (i.e., in a static environment), the quality impairment is also very small. The quality impairment significantly increases when the bitrate and the vibration level increase. For example, when the vibration level increases from 2 to 6, the quality impairment grows from 0.049 to 0.184 for 1.5 Mbps videos, and the impairment grows from 0.174 to 0.549 for 5.8 Mbps videos. With the least squares regression method, we can get the fitted surface, where the parameters c_3, c_4, c_5 are shown in Table 3.

3.3 Power Model

To model the power consumption during video streaming, we collect real measurement data by watching a short video "Everything Wrong With Transformers" from Youtube with different bitrates (i.e., 144p, 240p, 360p, 480p, 720p and 1080p) at different signal strengths. We use a rooted LG Nexus 5X smartphone running Android 7.0, which uses T-Mobile LTE network. Tcpdump is used to collect the network trace, and the Monsoon power monitor is used to measure the power level during video streaming. Since the battery connectors on the Nexus 5X smartphone are very tiny, it is very challenging to connect them to the power monitor. To solve this problem, we design a battery interceptor based on Flex PCB, which is a very thin circuit board that can be easily bent or flexed. The interceptor is connected with the smartphone's mainboard and the battery through the corresponding battery connector, and uses a customized circuit to modify the battery connection. As shown in Fig. 4(a), we use this interceptor to connect the smartphone with the Monsoon power monitor, which directly provides power supply for the smartphone. We also collect the wireless signal strength during video streaming by using an Android ADB shell *dumpsys telephony.registry* at the background. Using the collected data traces, we build two different power models based on whether there is data transmission (video downloading) or not.

TABLE 4
The power models.

State	Power (mW)
Without data trans.	$P_b(b) = 1121.5 + 24.71b$
With data trans.	$P_t(b,s) = 2301.2 + 439.6b$ $-41.57b^2 - 2.96s - 0.047s^2$

When there is no data transmission, i.e., the smartphone plays the buffered video segment, the power consumption (denoted by P_b) is only affected by the video bitrate. For high bitrate video, more data will be processed such as decoding and rendering, and hence more power will be consumed. Since P_b increases with the video bitrate, P_b is modeled as a linear function of the video bitrate. With the least squares regression method, we can get the fitted curve as shown in Fig. 4(b).

With data transmission, the power consumption (denoted by P_t) is affected by both video downloading and video processing, and hence it is affected by the bitrate (b) and the wireless signal strength (s). To model the relationship between power consumption and these two factors, we use the quadratic function [23] for curve fitting. Fig. 4(c) shows the fitted surface, where the R-square value of the fitting is 0.9866, which means a very high accuracy. The results are summarized in Table 4, where the video bitrate b is in terms of Mbps, the signal strength s is in terms of dBm and the power level is in mW.

Energy Consumption of Task T_i^j. Based on our power models, we can calculate the energy consumption of video streaming which consists of a set of tasks. We calculate the energy of task T_i^j based on if there is rebuffering.

Case (a) (No rebuffering): When there is no rebuffering, the energy consumption of task T_i^j depends on the relationship among L, β, B_i, and the downloading time $\frac{S_i}{R_i}$, where S_i is the segment data size and R_i is the downloading throughput.

If the video length of the downloaded but not yet viewed video after downloading the i^{th} segment is less than the buffer threshold β, i.e., $(B_i - \frac{S_i}{R_i} + L) < \beta$, the video player

starts to download the next segment when T_i^j is completed. In this case, the energy of task T_i^j is the energy consumed during the period of downloading the i^{th} segment.

If the buffered data after downloading the i^{th} segment reaches β, i.e., $(B_i - \frac{S_i}{R_i} + L) \geq \beta$, the player stops the downloading process and waits until the buffered video is less than β before downloading the next segment. In this case, the energy of task T_i^j includes the energy consumed during the period of downloading the i^{th} segment, and the energy consumed during the waiting period before starting to download the next segment.

Since the buffered data when the player downloads the i^{th} segment is B_i, there are $k = \lceil B_i/L \rceil$ video segments in the buffer, where the $(i-k)^{th}$ video segment is being played. Thus, we use $P_t(b_{i-k}, s)$ to calculate the energy consumed during the period of downloading the i^{th} segment. If more than one video segments will be played during the downloading period, since the power consumption P_t varies with video bitrate, we calculate the energy consumed during the downloading period by summing up the energies consumed when different bitrate videos being played during the downloading period. Since the waiting time is less than L, i.e., the $(i-k)^{th}$ video segment is being played during the waiting period, we use $P_b(b_{i-k})$ to calculate the energy consumed during the waiting period. P_t and P_b can be calculated based on Table 4.

In summary, the energy of task T_i^j when there is no rebuffering can be calculated with Equation 8, where $(x)_+ = max\{x, 0\}$.

$$\begin{aligned} E_a(T_i^j) &= P_t(b_{i-k}, s)(\frac{S_i}{R_i} - (\lceil S_i/R_i/L \rceil - 1)L) \\ &+ \sum_{g=1}^{\lceil S_i/R_i/L \rceil - 1} P_t(b_{i-k+g}, s)L \\ &+ P_b(b_{i-k})(B_i - \frac{S_i}{R_i} + L - \beta)_+ \end{aligned} \tag{8}$$

Case (b) (Rebuffering): When there is rebuffering, i.e., $B_i < \frac{S_i}{R_i}$, we can divide the video downloading into two parts: B_i and $\frac{S_i}{R_i} - B_i$. During the first part, the buffered video data is played out; while there is no video playback during the second part, i.e., rebuffering. We calculate the energy of T_i^j by summing up the energy consumptions of these two parts. Since there is no video playback during the rebuffering, we use $P_t(0, s)$ to calculate the energy consumed by downloading video during the rebuffering. Thus, the energy consumption of T_i^j when there is rebuffering can be calculated with Equation 9.

$$\begin{aligned} E_b(T_i^j) &= P_t(b_{i-k}, s)(B_i - (k-1)L) \\ &+ \sum_{g=1}^{k-1} P_t(b_{i-k+g}, s)L \\ &+ P_t(0, s)(\frac{S_i}{R_i} - B_i) \end{aligned} \tag{9}$$

In summary, the energy of task T_i^j can be calculated with Equation 10.

$$E(T_i^j) = \begin{cases} E_a(T_i^j), & \text{if } \frac{S_i}{R_i} \leq B_i \\ E_b(T_i^j), & \text{otherwise} \end{cases} \tag{10}$$

3.4 Problem Formulation

In this subsection, we formalize the context-aware and energy-aware video streaming problem. To determine the right bitrate for each video segment (task), we introduce a binary variable η_{ij} for bitrate selection, where $\eta_{ij} = 1$ if the video segment in task T_i is encoded with bitrate index j, i.e., T_i^j; otherwise, $\eta_{ij} = 0$. Since only one bitrate is selected for each task, $\sum_{j=1}^{V} \eta_{ij} = 1$.

In our context-aware and energy-aware video streaming, the goal is to minimize the energy consumption and maximize the QoE, and this can be achieved by selecting the right bitrate for each video segment downloaded in each task. This is a multi-objective optimization problem, and we apply the weighted sum method [24] to formalize it. For task T_i^j, the QoE ($Q(T_i^j)$) can be calculated with Equation 1, and the energy consumption ($E(T_i^j)$) can be calculated with Equation 10. Then, the optimization problem can be formulated as follows.

$$\begin{aligned} \textbf{min} \quad & \sum_{i}^{n} \sum_{j}^{V} \eta_{ij} (\gamma \frac{E(T_i^j)}{E(T_i^V)} - (1-\gamma) \frac{Q(T_i^j)}{Q(T_i^V)}) \\ \textbf{s.t.} \quad & \sum_{j=1}^{V} \eta_{ij} = 1, \ for \ \forall i. \end{aligned} \tag{11}$$

The objective of this optimization problem is to minimize energy and maximize QoE under the constraint that only one bitrate is selected for each task, $\sum_{j=1}^{V} \eta_{ij} = 1$. Since $E(T_i^j)$ and $Q(T_i^j)$ are measured using different units, they are normalized with the highest bitrate (i.e., $E(T_i^V)$ and $Q(T_i^V)$). γ is a weighting factor. With a smaller γ, the optimization problem puts more weight on maximizing QoE; with a larger γ, minimizing energy is more important.

4 Context-Aware and Energy-Aware Video Streaming

In this section, we present our context-aware and energy-aware video streaming algorithms. We first present an optimal algorithm which requires knowledge of all future tasks. Since it is impossible to have such knowledge in practice, we present an online bitrate selection algorithm by removing such assumption. In addition, we propose a crowdsourcing based bitrate selection algorithm.

4.1 The Optimal Algorithm

For a video streaming consisting of n tasks, our goal is to find bitrate decisions to minimize the energy and maximize the QoE. The process of determining the right bitrate for each video segment (each task) can be mapped to the shortest path problem [10], as shown in Fig. 5.

Let node T_s and T_e denote the start and end of the video streaming process, respectively. Since there are V bitrates available, we add V nodes for each task (T_i), where T_i^j corresponds to the video segment downloaded in T_i and it is encoded with bitrate index j. As shown in Fig. 5, we add edges from T_s to each node of T_1, and from each node of T_n to T_e. We add an edge from each node of task T_i to all V nodes of the next task. By considering both energy consumption and QoE, the weight of an edge is defined as

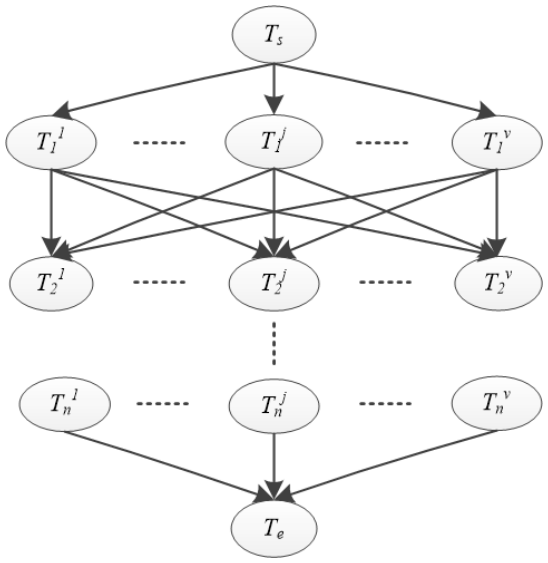


Fig. 5. Mapping the context-aware and energy-aware video streaming problem to the shortest path problem.

$(\gamma \frac{E(T_i^j)}{E(T_i^V)} - (1-\gamma)\frac{Q(T_i^j)}{Q(T_i^V)})$. For edges from nodes of T_n to T_e, their weights are defined as 0.

In this graph, we consider all bitrate cases of each task and all possible schedule paths between tasks, so each path from T_s to T_e maps to a bitrate selection sequence for the video streaming, and vice versa. As a result, the shortest path from T_s to T_e corresponds to the optimal bitrate selection of all tasks in the video streaming process such that the QoE is maximized and the energy is minimized.

Based on the graph, we can use the Dijkstra's algorithm to find the shortest path. Given a video streaming process consisting of n tasks, the graph has $O(nV)$ nodes and $O(nV^2)$ edges. As the time complexity of the Dijkstra's algorithm is $O((N+E)logN)$, where N is the number of nodes and E is the number of edges, the time complexity of the optimal algorithm is $O(nV^2log(nV))$.

4.2 The Online Bitrate Selection Algorithm

Since the optimal algorithm requires the complete knowledge of all future tasks, which is impossible in practice, it can only be served as a performance upper bound. In this subsection, we propose an online bitrate selection algorithm, which first estimates the available bandwidth and the vibration level, and then determines the right bitrate for each video segment.

To estimate the network bandwidth, similar to [2], we use the harmonic mean of the downloading throughput of the past several segments to estimate the network bandwidth. Since the network condition varies widely during video streaming, especially when the user is on a moving vehicle, some downloading throughput can be much higher or lower than others among the past several segments. The harmonic mean is used to eliminate the impacts of these fluctuations.

The vibration level is estimated based on the collected accelerometer data in a time window from the current time to the previous $0.2 * \beta$, where β is small, i.e., 30 seconds. This vibration level is used to determine the right bitrate for the video segment. Since the player buffer threshold (β) is very small, the downloaded video segment will be played after very short time. Since the time interval between the video downloading and the video playback is very small, the vibration level when downloading video can be used to estimate the vibration level when the downloaded video is played.

Algorithm 1: The Online Bitrate Selection Algorithm

Input : L, β, γ, B_i, previous bitrate j_{i-1}
Output: j_i: bitrate level for segment i

1 $\widehat{R}_i \leftarrow$ estimated bandwidth
2 $\widehat{v}_i \leftarrow$ vibration calculated with Eq. (5)
3 $j_i^\star \leftarrow \text{argmin}_{j\in\{1,...,V\}}(\gamma \frac{E(T_i^j)}{E(T_i^V)} - (1-\gamma)\frac{Q(T_i^j)}{Q(T_i^V)})$
4 **if** $j_i^\star > j_{i-1}$ **then**
5 $\quad j_i^\star \leftarrow j_{i-1} + 1$
6 **else if** $j_i^\star < j_{i-1}$ **then**
7 $\quad j_i^\star \leftarrow \max\{j | j \in \{j_i^\star, ..., j_{i-1}\}$ and $(\frac{S_i^j}{\widehat{R}_i} \le B_i)\}$
8 $\quad //S_i^j$: data size of segment i at bitrate j
9 **end**
10 **return** $j_i^\star$

Based on the estimated bandwidth and the vibration level, the online bitrate selection algorithm can calculate the energy consumption ($E(T_i^j)$) and the user perceived QoE ($Q(T_i^j)$) when downloading the i^{th} video segment encoded with bitrate j. To minimize energy and maximize QoE, we apply the same objective function of Equation 11, i.e., $\gamma \frac{E(T_i^j)}{E(T_i^V)} - (1-\gamma)\frac{Q(T_i^j)}{Q(T_i^V)}$.

The online bitrate selection algorithm is described in Algorithm 1. Due to network variations, a sudden large bitrate increase may result in more rebuffering events and frequent bitrate changes, and a sudden large bitrate drop may lead to severe QoE impairment. To deal with these problems, the online algorithm first computes a *reference* bitrate for the i^{th} video segment (line 4) based on Equation 11. Then, it determines the final bitrate, based on the relationship between the reference bitrate and the bitrate of the previous video segment.

If the reference bitrate is higher than that of the previous video segment, the algorithm does not immediately jump to the reference bitrate, as the reference bitrate can be several levels higher than that of the previous segment. Instead, for each bitrate increase, the algorithm selects the bitrate which is one level higher than the bitrate of the previous video segment (line 5-6). This gradual bitrate change reduces the QoE impairment caused by frequent bitrate changes due to network variations. If the network bandwidth is consistently high, i.e., the reference bitrate is higher than the bitrate of the previous video segment for consecutive video segments, the video bitrate will gradually increase to the reference bitrate.

If the reference bitrate is lower than that of the previous video segment, the algorithm does not immediately drop to the reference bitrate to reduce the QoE impairment. Instead, for each bitrate decrease, the online algorithm searches from the bitrate of the previous video segment to the reference bitrate, and finds the first bitrate which can be used to suc-

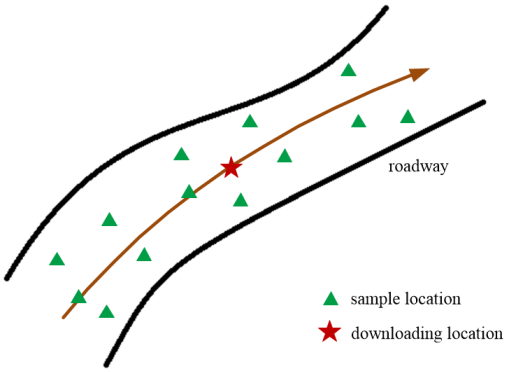


Fig. 6. Crowdsourcing based prediction.

cessfully download the video segment before the buffered data is drained out (line 7-9). If the network bandwidth is consistently low, i.e., the reference bitrate is lower than the bitrate of the previous video segment for consecutive video segments, the video bitrate will eventually decrease to the reference bitrate.

4.3 The Crowdsourcing Based Algorithm

The proposed online bitrate selection algorithm can estimate the available bandwidth and the vibration level, and then determine the right bitrate for each video segment; however, it underperforms the optimal algorithm which has the complete knowledge of all future tasks. For example, on a moving vehicle, the network condition may dramatically change during some time period. Then, using the downloading throughput of the past several segments measured long time ago for bandwidth prediction may not be accurate. Similar problems exist for vibration prediction. To further improve the performance of the online algorithm, we propose a crowdsourcing based approach for bandwidth and vibration prediction; i.e., bandwidth and vibration value of each location can be predicted using the downloading throughput and vibration values collected from other users at (or around) this location. Although crowdsourcing based approaches have been proposed for video streaming [15], [16], most of them focus on maximizing QoE, and none of them considers energy and context issues.

In crowdsourcing, mobile devices collect data tuples such as bandwidth, vibration, and capture time for each location during video streaming, and then upload these data to the server. Since the crowdsource data (e.g., along the commute route) usually do not frequently change, the crowdsourced data can be downloaded to the client beforehand. In a streaming session, the client can quickly obtain the crowdsourced data base on the current location and perform bandwidth and vibration prediction. Based on the predicted bandwidth and vibration, the client determines the best quality level for the video segment to be downloaded. Next, we first describe the data collection and query process, the bandwidth and vibration prediction method, and then present the crowdsourcing based bitrate selection algorithm.

Data Collection and Query

The server maintains the records of previously measured bandwidth and vibration values associated with their locations (i.e., GPS coordinates), which can be uploaded by the clients during video streaming. The data tuple for each location is represented as $< loc, time, bw, vib >$, where loc is the GPS coordinate, $time$ is the timestamp when data is recorded, bw is the downloading throughput, and vib is the vibration value which can be calculated with Equation 5.

When downloading a video segment, the client queries the server for the bandwidth and vibration values by sending its current location and moving speed to the server. To reduce the query delay and save energy, we propose the following solution. Since the crowdsourced data usually do not frequently change, the client can download such data daily (or several days) when charging at home, similar to travelers downloading google map beforehand to avoid using data plan. The data size should not be too large since each data tuple only takes several bytes, and the client only needs to download data in one area, e.g., along the commute route. Then, the query will be very simple, similar to offline Google map, where the client queries are based on the current location and immediately obtain the bandwidth and vibration level.

In a video streaming session, the network bandwidth and vibration level for downloading each video segment can be predicted based on the crowdsourced data, as detailed in the following subsection.

Bandwidth and Vibration Prediction

Fig.6 illustrates the concept of bandwidth and vibration prediction. In the figure, green triangles (called sample locations) represent the locations where bandwidth and vibration values of other users (bus or train riders) were collected, and the red star (called downloading location) represents the location where the video segment will be downloaded. Since bandwidth and vibration values do not change too much within a small area, the bandwidth and vibration values of the current user can be estimated using those of adjacent locations. We use the bandwidth and vibration values of the K nearest sample locations to make prediction; i.e., the weighted average of the values at the sample points. Since the values at the sample locations close to the downloading location are more relevant than those far away, we assign larger weight to closer samples. More specifically, we apply an inverse distance weighted (IDW) interpolation [25] for the estimation. IDW assigns greater weights to the values of the sample locations closer to the downloading location, and the weights diminish as a function of distance. The bandwidth (or vibration) value R at the downloading location l can be calculated with Equation 12.

$$R = \frac{\sum_{i=1}^{K} w_i \cdot R_i}{\sum_{i=1}^{K} w_i} \tag{12}$$

where R_i the network bandwidth at the sample location l_i, $w_i = \frac{1}{d(l_i,l)^p}$ is the weight, $d(l_i, l)$ represents the distance between the sample location l_i and the downloading location l, and p is the power parameter that determines the rate at which the weights decrease. We set $p = 2$

Algorithm 2: The Crowdsourcing Based Bitrate Selection Algorithm

Input : L, β, γ, B_i, previous bitrate j_{i-1}
Output: j_i: bitrate level for segment i

$\widehat{R}_{[i,i+w-1]} \leftarrow$ bandwidth for segments $[i, i+w-1]$
$\widehat{v}_{[i,i+w-1]} \leftarrow$ vibration for segments $[i, i+w-1]$
for $t \leftarrow i$ ***to*** $i + w$ *-1* **do**
  for $j_t \leftarrow 1$ ***to*** V **do**
    $U(\Phi_t, j_t) = \min_{j_{t-1} \in \{1,\ldots,V\}} \{U(\Phi_{t-1}, j_{t-1}) + \mathcal{O}(\Phi_{t-1}, j_{t-1})\}$
    // record the optimal solution
    $prev(\Phi_t, j_t) = j_{t-1}$
    // update system state
    $\Phi_t = \mathbf{f}(\Phi_{t-1}, \widehat{R}_{t-1}, \widehat{v}_{t-1}, j_{t-1})$
  end
end
Backtrack from $prev(\Phi_{i+w-1}, j_{i+w-1})$ to find the optimal j_i
return j_i

for the experiments. Since network bandwidth varies with time and location, bandwidth prediction using only the crowdsourced data (which may be scarce at some locations) may not reflect the real network condition. That is, for downloading video segment at some locations, bandwidth prediction may be inaccurate if we use the bandwidth samples far away. Similarly, bandwidth prediction may be inaccurate if we use the bandwidth samples collected at a different time period (i.e., predicting bandwidth at day time using samples collected at midnight).

To address this issue, we only use the crowdsourced data within a predefined spatial region (e.g., 1000 square meters) and a time period (within two hours). Then, we use the downloading throughput of the past K segments to calibrate the prediction. Specifically, we use the weighted average network throughput, which is expressed as Equation 13.

$$\widehat{R}_i = \omega_i \widehat{R}_i^s + (1 - \omega_i)\widehat{R}_i^t \tag{13}$$

where $\widehat{R}_i^s$ is the estimated bandwidth calculated using the K nearest crowdsourced bandwidth samples, $\widehat{R}_i^t$ is the estimated bandwidth calculated using the past K downloading throughput, and ω_i indicates the confidence of using the crowdsourced data to estimate the network bandwidth. The calibrating factor ω_i is adaptively updated and is defined as $\omega_i = \frac{1}{1+|\widehat{R}_{i-1}^s - R_{i-1}|/R_{i-1}}$, where R_{i-1} denotes the throughput when downloading task T_{i-1}. With a larger ω_i value, i.e., $\widehat{R}_{i-1}^s$ is close to R_{i-1}, we put more weight on $\widehat{R}_i^s$.

Crowdsourcing Based Bitrate Selection Algorithm

In our online bitrate selection algorithm, the bitrate is selected based on the predicted bandwidth and vibration level for the next video segment. Due to large bandwidth and vibration variations, the online algorithm may select a bitrate too big or too small, and then affect the QoE. One nature solution is to consider several video segments in the future when making bitrate selections. However, this is hard for the online algorithm, where the bandwidth prediction becomes less accurate for future video segments. With crowdsourcing based solution, we can predict the bandwidth and vibration level of multiple video segments in the future, based on which we can choose a better bitrate. To achieve this goal, we apply an Model Predictive Control (MPC) based optimization framework [20], which is widely used to optimize a complex control objective in a dynamical system under constraints.

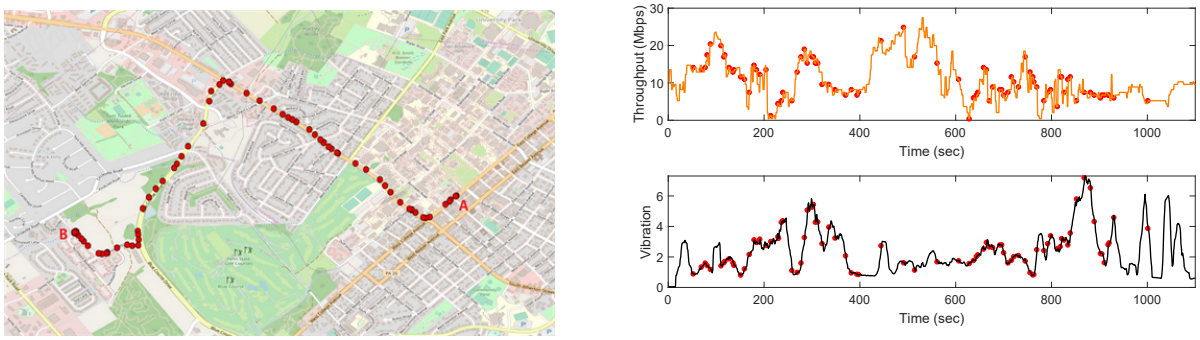
In MPC, the video streaming process is modeled as a discrete-time dynamical system, where the system state when downloading video segment i is defined by the buffered data B_i and the bitrate level for previous segment j_{i-1}; i.e., $\Phi_i =< B_i, j_{i-1} >$. The system state evolves based on the video bitrate selected for segment i; i.e., the buffered video when downloading video segment $i+1$ becomes $B_{i+1} = max(B_i - \frac{S_i}{R_i}, 0) + L - \triangle t_i$, where S_i is the data size of video segment i encoded at bitrate j_i, L is the length of the video segment, and $\triangle t_i$ is the waiting time. If the buffered video data reaches the buffer threshold β, the player waits for $\triangle t_i$ before requesting segment $i+1$; i.e., $\triangle t_i = max(max(B_i - \frac{S_i}{R_i}, 0) + L - \beta, 0)$. The system dynamics are summarized as $\Phi_{i+1} = \mathbf{f}(\Phi_i, R_i, v_i, j_i)$.

For a state transition from Φ_i to Φ_{i+1} (denoted as $\Phi_i \xrightarrow{j_i} \Phi_{i+1}$), the objective is to minimize energy and maximize QoE as shown in Equation 11; i.e., $\mathcal{O}(\Phi_i, j_i) = \gamma \frac{E(T_i^{j_i})}{E(T_i^V)} - (1-\gamma)\frac{Q(T_i^{j_i})}{Q(T_i^V)}$. We use MPC to optimize the aggregate objective of the next several video segments. Specifically, given the predicted bandwidth $\widehat{R}_{[i,i+w-1]}$ and vibration level $\widehat{v}_{[i,i+w-1]}$ during the next w video segments, MPC is used to optimize the aggregate objective of these w video segments. The MPC based optimization problem can be expressed as follows.

$$\begin{aligned} \textbf{min} \quad & \sum_{t=i}^{i+w-1} \mathcal{O}(\Phi_t, j_t) \\ \textbf{s.t.} \quad & \Phi_{t+1} = \mathbf{f}(\Phi_t, \widehat{R}_t, \widehat{v}_t, j_t) \end{aligned} \tag{14}$$

where w is the optimization window size. It searches the bitrate levels $j_{[i,i+w-1]}$ that maximize the aggregate objective within the optimization window, and then requests video segment i encoded at bitrate level j_i. In the next interval, the optimization window is moved forward to $[i+1, i+w]$ to determine the video bitrate for segment $i+1$. This bitrate selection process repeats for the remaining video segments.

Although a brute force search can find an optimal solution for Equation 14, its computational complexity is $O(V^w)$, where V is the number of bitrate levels. There will be a state explosion problem for large w. To efficiently find the solution for Equation 14 in practice, we propose a dynamic programming based approach. Based on the fact that the accumulative objective at segment $i+1$ depends on the solution at segment i and the transition $\Phi_i \xrightarrow{j_i} \Phi_{i+1}$, the Bellman equation for the dynamic programming is $U(\Phi_{i+1}, j_{i+1}) = \min_{j_i \in \{1,\ldots,V\}} \{U(\Phi_i, j_i) + \mathcal{O}(\Phi_i, j_i)\}$, where $U(\Phi_i, j_i)$ is the optimal utility up to segment i with bitrate level j_i. The algorithm is described in Algorithm 2, which computes the optimal solution for each segment within the optimization window based on the optimal solution to the previous segment (i.e., the Bellman equation). After the algorithm is terminated, the right bitrate j_i for segment i can be



(a) (b)

Fig. 7. (a): Video streaming on a moving vehicle along the route from location A to B, where each dot represents the location a downloading task happens. (b): The throughput and vibration traces.

TABLE 5
Video traces.

Video ID	Length (sec)	Data size (MB)	Avg. vibration
1	198	65.1	6.83
2	371	123.8	2.46
3	449	140.6	6.61
4	498	152.2	6.41
5	612	173.1	5.23

computed by back tracking the solution selected for the last segment in the optimization window.This will reduce the time complexity to $O(wV)$.

5 Performance Evaluations

In this section, we evaluate the performance of the proposed algorithms through extensive trace-driven simulations and compare it with other existing algorithms.

5.1 Experiment Setup

We collect traces by watching videos from Youtube using LG Nexus 5X smartphone. Subjects watch five videos with various video length and data size, and under different contexts, as shown in Table 5. We collect three kinds of traces: the network trace by using tcpdump to extract the downloading time and the downloading data size, the signal strength trace by using a ADB shell, and the accelerometer data in the smartphone. For simulations, each video is cut into 2 seconds segments and is encoded with fourteen bitrates, i.e., $\{0.1, 0.2, 0.24, 0.375, 0.55, 0.75, 1.0, 1.5, 2.3, 2.56, 3.0, 3.6, 4.3, 5.8\}$ Mbps. We set the buffer threshold $\beta = 30$ seconds. We equally consider minimizing energy and maximizing QoE; i.e., $\gamma = 0.5$.

Based on these traces, we compare the performance of the following approaches.

- `Youtube`: Video streaming with the original Youtube app at a bitrate of 5.8 Mbps (i.e., with resolution of 1080p).
- `FESTIVE [2]`: A throughput-based bitrate adaptation approach, which uses the harmonic mean of the past several throughput measurements to estimate the available bandwidth, and then selects the highest available bitrate that is just below the estimated bandwidth.
- `BBA [26]`: A buffer-based bitrate adaptation approach. BBA uses throughput to control video bitrate at the startup phase. After reaching the steady state, BBA maps the current buffer level to bitrate selection using a linear function.
- `OBA`: The online bitrate selection algorithm, which selects the bitrate that minimizes energy and maximize QoE.
- `CBA`: The crowdsourcing based bitrate selection algorithm, which selects the bitrate that minimizes energy and maximize QoE. Different from OBA, it uses crowdsourced data for bandwidth prediction and vibration prediction.
- `Optimal`: The optimal algorithm, which has the complete knowledge of all future tasks.The Optimal algorithm is impossible to achieve in practice, so it only provides a performance upper bound.

5.2 Performance of Crowsourcing Based Bandwidth and Vibration Prediction

As shown in Fig. 7, we use the bandwidth and vibration traces measured when watching video on a moving vehicle to evaluate the performance of the crowdsourcing based approach for bandwidth and vibration prediction. On the same route from location A to B, as shown in Fig. 7(a), we conducted multiple experiments to collect traces by watching videos from Youtube. The network traces are collected by using Tcpdump to extract the downloading time and the downloading data size. All packets with an interval less than one second are considered as the same downloading task. Fig. 7(a) shows an example of the downloading tasks for one experiment, where each dot in the figure represents the location where a video downloading task takes place. Fig. 7(b) shows the bandwidth and vibration traces for the experiment in Fig. 7(a), where red dots on the traces correspond to the downloading tasks in Fig. 7(a).

For the two proposed algorithms (OBA and CBA), the performance is affected by the bandwidth and vibration prediction, where the best (worst) prediction scenario will be the best (worst) performance scenario for the approach. Fig. 8 compares the prediction result of the crowdsourcing based approach (i.e., OBA) and the harmonic mean approach (i.e., CBA). As can be seen, the crowdsourcing based approach

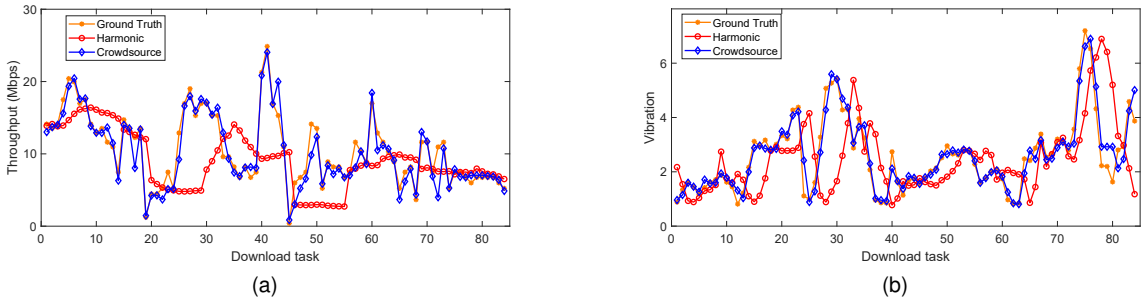


Fig. 8. (a): Throughput prediction. (b): Vibration prediction.

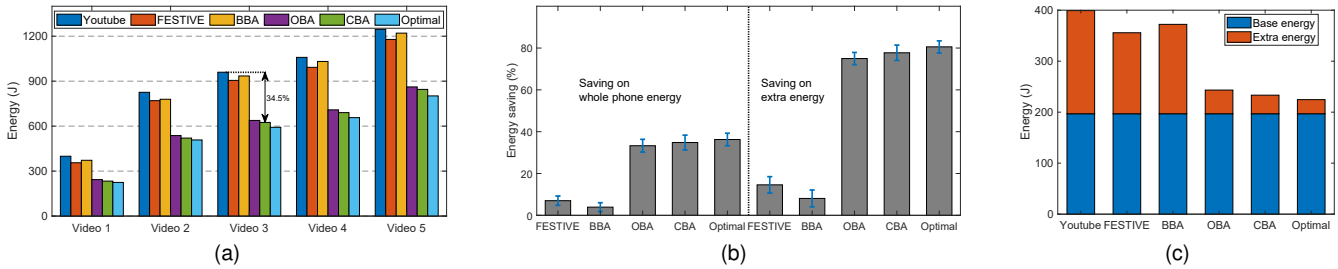


Fig. 10. (a): Comparison of different approaches on energy consumption. (b): Energy saving compared to Youtube. (c): The base energy and the extra energy for trace 1.

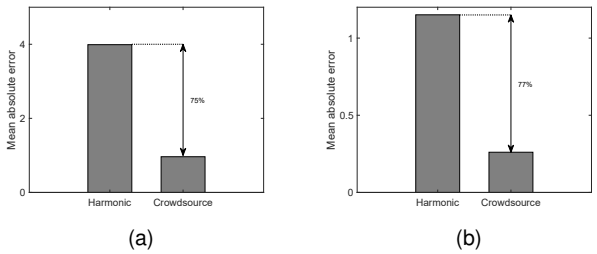


Fig. 9. Mean absolute error for bandwidth prediction (a) and vibration prediction (b).

can react to network variations better than the harmonic mean approach. For example, the harmonic mean approach underestimates the network bandwidth from the 26^{th} to 32^{th} downloading task, leading to downloading video segments at low quality. In contrast, when the network condition drops from the 33^{th} to 39^{th} downloading task, the harmonic mean approach overestimates the network bandwidth by using the downloading throughput (that is high) of the past segments. As a result, rebuffering events may take place if the client keeps requesting video segments encoded at high bitrate that is much greater than the network capacity. Fig. 9 compares the Mean Absolute Error (MAE) of these two approaches. MAE is the error calculated as an average of the absolute difference between the predicted value and the ground truth; i.e., $MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \widehat{y}_i|$, where $\widehat{y}_i$ is the prediction and y_i is the true value. As can be seen, compared to the harmonic mean approach, the crowdsourcing based approach can improve bandwidth prediction by up to 75% and 77% for vibration prediction.

5.3 Energy Comparison

The energy consumption is calculated based on the power models shown in Section 3.3. As shown in Fig. 10(a), we compare the energy consumption of different approaches based on the measured traces. As can be seen in the figure, compared to Youtube which downloads video segments at bitrate of 5.8 Mbps and consumes the most energy, other five approaches can save energy due to the bitrate adaptations. FESTIVE always downloads video segments encoded at the highest available bitrate that is just below the estimated bandwidth, and consumes much energy. BBA is more aggressive to download higher bitrate segments after the buffer reaches the steady state, i.e., BBA requests the highest bitrate after the buffered data is larger than the predefined upper threshold, thus consumes much more energy compared to FESTIVE. Compared to FESTIVE and BBA, OBA can save more energy, since OBA takes into account the context of video streaming and download low bitrate videos when the vibration level is high. As shown in Fig. 10(b), OBA can save 33.3% energy on average, which is very close to Optimal (36.2%), and is much higher than FESTIVE (7%) and BBA (3.9%). By using the crowdsourcing based approach for bandwidth and vibration prediction, CBA can reduce the energy consumption by 34.8%.

As shown in Fig. 10(c), we separate the total energy consumption during video streaming into two parts: base energy and extra energy consumption. The base energy is the energy consumed when all video segments are encoded with the lowest bitrate, which includes the energy consumed by the screen and the energy consumed by data transmission and video processing. Thus, the base energy is the minimum energy consumption for video streaming. All video streaming approaches will try to choose bitrates higher than the lowest bitrate to improve QoE, at the cost of more energy consumption. The extra energy is the energy difference between the energy consumed by the selected video streaming approach and the base energy. As shown in Fig. 10(b), when only considering the extra energy, CBA can

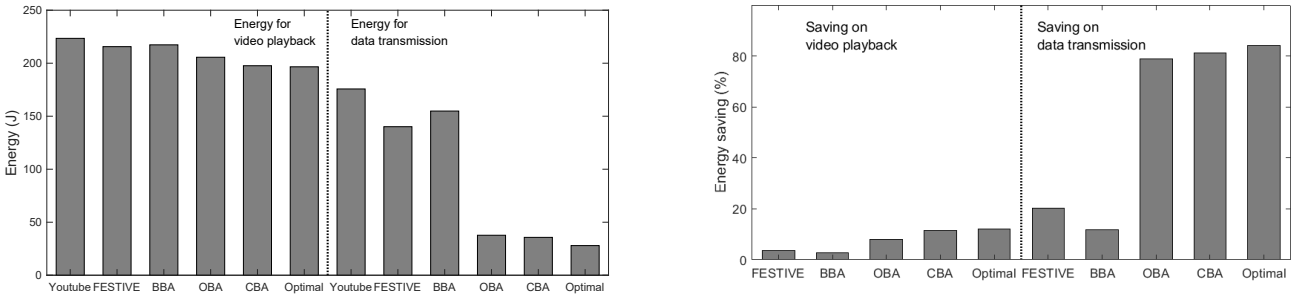


Fig. 11. (a): Energy consumption for video playback and data transmission. (b): Energy saving compared to Youtube.

TABLE 6
The power model validation.

Bitrate (Mbps)	Measured energy (J)	Calculated energy (J)	Error ratio
5.8	708.13	713.59	0.77%
3.0	648.69	658.62	1.53%
1.5	637.36	622.55	2.32%
0.75	615.69	609.79	0.96%
0.375	608.04	597.75	1.69%
0.1	597.02	589.38	1.28%

save 78.7% energy which is very close to Optimal (80.5%), and it is much higher than FESTIVE (14.6%) and BBA (8.1%). For smartphones with smaller screen size, the base energy part will be smaller and then the total energy consumption will be smaller. Relatively speaking, the energy saving will be more significant since the denominator is smaller.

5.3.1 Video Playback vs. Data Transmission

To further analyze the energy consumption, we separate the total energy consumption during video streaming into two parts: energy consumption for data transmission and energy consumption for video playback. For each video, we first identify the download periods and the signal strength during each download period from the collected data traces. We then calculate the energy consumption using the power models shown in Section 3.3. As shown in Table 4, $P_t(b, s)$ indicates the power consumption affected by both data transmission and video playback, and $P_b(b)$ indicates the power consumption during video playback only. We calculate the energy consumption for data transmission as $P_t(b_1, s) - P_b(b_2)$ times the duration of download periods, where b_1 and b_2 are the bitrate of the video segment being downloaded and the bitrate of the video segment being played during the video downloading period, respectively. Then the energy consumption for video playback during the whole video streaming process can be calculated as the energy difference between the total energy consumption and the energy consumption for data transmission.

Taking video 1 as an example, the energy consumption for data transmission and video playback is shown in Fig. 11. As can be seen in Fig. 11(a), Youtube consumes the most energy for video playback compared to other five approaches. This is because, for high bitrate video, more data will be processed such as decoding and rendering, and hence more energy will be consumed. Compared to Youtube, as shown in Fig. 11(b), the energy saving for video playback is 8% for OBA, which is twice (three times) of that for FESTIVE (BBA). The energy saving will be more significant if we compare the energy consumption for data transmission. As can be seen in Fig. 11(b), compared to Youtube, CBA can save up to 81.3% energy for data transmission, which is close to Optimal (84.1%), and it is much higher than FESTIVE (20.2%) and BBA (11.9%). This is because the power consumption of the wireless interface is high when it is turned on during data transmission, and is much lower when the interface is turned off. Approaches, such as Youtube, FESTIVE and BBA, request high bitrate video segments and consume more energy, since the wireless interface stays on for long time to download large amount of data.

5.3.2 Power Model Validation

To validate the power models, we compare the energy consumption measured by the Monsoon power monitor and the energy consumption calculated using the power models. For each video, we first identify the download periods from the packet trace obtained by Tcpdump, and obtain the signal strength during each download period from the signal strength trace. We then calculate the energy consumption using the power models shown in Section 3.3. Here we show an example when the signal strength is -90 dBm. The energy consumption for different bitrate videos based on the real power trace is shown in Table 6. The calculated energy consumptions are very close to the real measurements, which indicates that our power models are pretty accurate. The error ratio is consistently less than 3%, with an average of 1.43%. Note that the power model is dedicated for the specific smartphone used in this research and can be enhanced by considering other factors such as the current level of the battery.

5.4 QoE Comparison

Fig. 12(a) compares the QoE of all approaches. As can be seen, Youtube outperforms other approaches for all traces because Youtube requests all video segments at the highest bitrate, and it suffers no quality impairment from bitrate changes. However, the QoE gap between Youtube and others is very small. This is because the QoE for video streaming on smartphones does not improve too much when the video bitrate is very high, and the perceived quality is highly affected by the vibrating environment. Compared

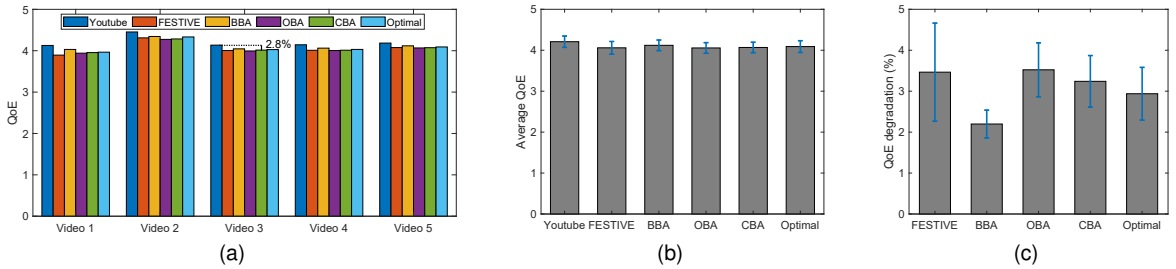


Fig. 12. (a): Comparison of QoE for each trace. (b): The average QoE for each approach. (c): QoE degradation compared to Youtube.

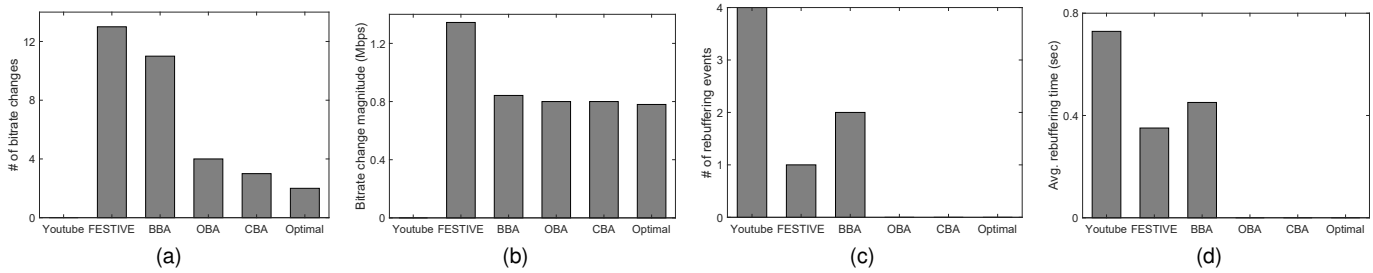


Fig. 13. (a): The number of bitrate changes. (b): The average magnitude of bitrate change. (c): The number of rebuffering events. (d): The average rebuffering duration.

to other traces, the QoE for trace 2 is much better for all approaches due to its low vibration level.

The overall QoE for all approaches is shown in Fig. 12(b). It can be seen that OBA can achieve very high QoE. This is because OBA considers the network bandwidth and the vibration level when selecting video bitrate for each video segment. OBA requests video segments encoded at the most suitable bitrate under the vibrating environment, and thus gains high QoE. After reaching the steady state, BBA is aggressive to request higher bitrate videos, and thus gains higher QoE at the cost of higher energy consumption. Fig. 12(c) shows the QoE degradation compared to Youtube. Compared to Youtube, the average QoE degradation of CBA is 3.2%, which is better than OBA (3.6%), and similar to FESTIVE (3.5%), Optimal (2.9%), and BBA (2.2%).

5.4.1 *Bitrate Changes and Rebuffering Events*

As described in Section 3.2, the quality impairment during data transmission includes the impact of rebuffering events and bitrate changes. Here we show an example of video 1. As can be seen in Fig. 13(a) and (b), Youtube requests all video segments at the same quality (i.e., the highest bitrate), and suffers no quality impairment from bitrate changes. FESTIVE generates the most number of bitrate changes due to the network fluctuations, because it always requests video segments encoded at the highest available bitrate that is just below the estimated bandwidth. BBA gradually increases video bitrate before reaching the steady state, and thus generates a number of bitrate changes. After the buffered data is larger than the predefined threshold, BBA always requests video segments at the highest bitrate and thus generates less number of bitrate changes compared to FESTIVE. In contrast, OBA generates much less number of bitrate changes (i.e., up to 70% less than that of FESTIVE) by considering the network conditions and the context of video streaming. OBA limits sudden bitrate changes; i.e., bitrate increase and then drop due to network fluctuations.

Figures 13(c) and 13(d) compare the rebuffering events of all approaches. Rebuffering event happens when the video segment cannot be successfully downloaded in time before the buffered video data drains out. As can be seen, Youtube generates the most number of rebuffering events, which is four times of FESTIVE and twice of BBA. This is because Youtube always downloads video segments encoded at the highest bitrate, and thus has the highest probability of encountering rebuffering events when the network condition suddenly varies. Compared to FESTIVE, BBA is more aggressive to download higher bitrate segments after reaching the steady state, and thus generates more number of rebuffering events. As can be seen, CBA and OBA do not generate any rebuffering events, since they request video segments encoded at the most suitable bitrate under the network conditions and the vibrating environment.

5.5 Impact of Parameter γ

For context-aware and energy-aware video streaming, our goal is to minimize energy and maximize QoE. As shown in Fig. 10 and Fig. 12, for trace 3, CBA save up to 34.5% energy at the cost of only 2.8% QoE degradation. We use the ratio of energy saving over QoE degradation to evaluate the overall performance, and the result is shown in Fig. 14. The weighting factor γ directly affects the objective of the optimization problem in Equation 11. To see the impact of γ on the performance, we run experiments with different γ values that ranges from 0.1 to 0.9, with the interval of 0.1. Note that parameter γ does not affect the bitrate decisions (i.e., the streaming performance) for FESTIVE and BBA. As can be seen in Fig. 14, the ratio decreases as γ increases. This is because with a smaller γ, the optimization problem puts more weight on maximizing QoE; i.e., the ratio becomes large when QoE degradation is very small. In contrast, with a larger γ, the optimization problem priorities minimizing energy consumption.

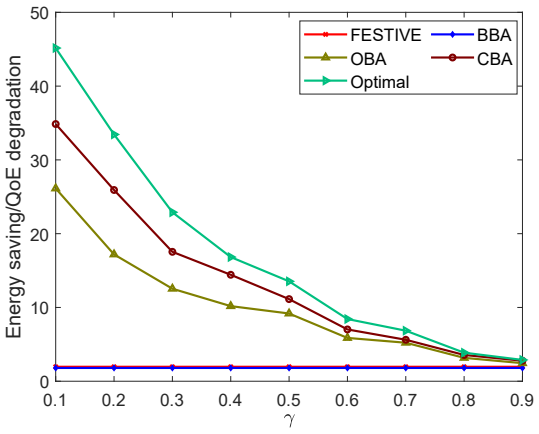


Fig. 14. The ratio of energy saving over QoE degradation.

Taking $\gamma = 0.5$ as an example, i.e., we equally consider minimizing energy and maximizing QoE. On average, OBA achieves better performance compared to FESTIVE (4.8X) and BBA (5.1X). This is because the QoE does not improve too much when the bitrate is very high for video streaming on smartphones and is highly affected by the vibrating environment, FESTIVE and BBA waste too much energy on maintaining high bitrate video streaming in the vibrating environment. As also can be seen, CBA can further improve OBA by twenty percent in terms of the ratio of energy saving over QoE degradation.

5.6 Comparisons of Vibration Based Schemes

In the previous subsections, we compared the performance of the proposed algorithms with others. In this subsection, we compare it to a simple vibration based approach (called VBA), which is based on a simple lookup table of vibration level to select the bitrate for video segments. Since each video is encoded with fourteen bitrates and the vibration level is less than seven most of time in our experiments, we can define the lookup table as a mapping function, i.e., $j = min(13, max(14-int(2*v), 0))$, where v is the vibration level and j is the bitrate level ($j = 13$ represents the highest bitrate and $j = 0$ represents the lowest bitrate).

Fig. 15 shows the energy saving and QoE degradation of OBA and VBA compared to Youtube. As can be seen, for trace 1, VBA can save 44.3% energy which is slightly better than that of OBA (with $\gamma = 0.5$). However, the QoE degradation of VBA (46.9%) is much larger than that of OBA (4.5%), since VBA downloads all video segments at very low bitrate (i.e., at an average bitrate of 0.212 Mpbs). By contrast, VBA consumes more energy than OBA for trace 2 where the vibration levels are low, because it always downloads high bitrate video segments. However, the QoE of VBA is lower than that of OBA due to high frequency of rebuffering events and bitrate changes. Compared to Youtube, OBA can save 33.3% of energy when all five traces are considered, at the cost of 3.6% QoE degradation, which is much better than that of VBA (i.e., 36.4% energy saving at the cost of 34.5% QoE degradation). Different form VBA, OBA selects bitrate considering both available bandwidth and vibration level. Moreover, it relies on optimization techniques to maximize QoE and minimize energy based on the network bandwidth

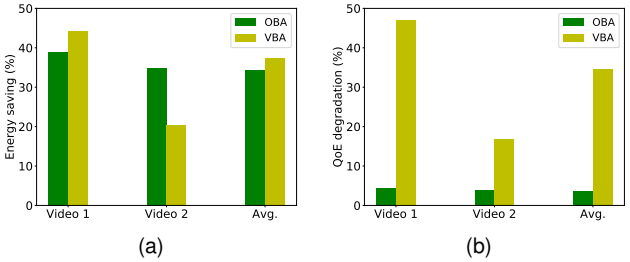


Fig. 15. Comparisons of vibration based schemes: (a) Energy saving compared to Youtube. (b) QoE degradation compared to Youtube.

and the vibration level. Thus, OBA can perform much better than VBA which simply maps vibration level to bitrate selection. Note that our CBA algorithm performs better than OBA, and then is much better than VBA.

6 Related Work

QoE. There has been considerable research on maximizing QoE for DASH based video streaming. Researchers [19], [27] have proposed to model QoE for DASH based video streaming with multiple metrics: average video bitrate, bitrate changes between successive segments, and rebuffering event. FESTIVE [2] balances stability and efficiency, and provides fairness among video players. Rather than estimating network bandwidth, researchers [5], [26] have also proposed to use the current buffer level to determine the video bitrate. Li *et al.* [28] proposed a probe and adapt bitrate adaptation scheme. Yin *et al.* [20] proposed a control-theoretic model, which optimizes the user perceived QoE by considering throughput and buffer information. In [29], a neural network was trained to select the video bitrate of future video segments. Balasubramanian *et al.* [30] proposed device-to-device (D2D) based caching techniques to minimize the latency for video streaming, and reinforcement learning techniques were designed in [31] to reduce the delay of downloading the video content in D2D networks. Enghardt *et al.* [32] proposed an application-informed approach to select the most suitable access network for each video segment. These methods try to maximize the QoE by considering the quality impairment caused by data transmission, and none of them considers the quality impairment caused by vibrations.

Energy Consumption. Researchers have proposed various techniques [7], [9], [33] to reduce the energy consumption of the wireless interface during video streaming. Hu *et al.* [7] proposed techniques to save energy based on whether the user tends to watch video for a long time, skip, or early quit. Hoque *et al.* [9] designed a download scheduling algorithm base on crowd-sourced viewing statistics. Wu *et al.* [33] designed an energy efficient video streaming scheme over heterogeneous networks. There has been some research [8], [34], [35] on reducing the tail energy in cellular networks during data transmission. Researchers have also proposed techniques [10], [11], [12], [13], [36], [37] to reduce the energy consumption of video processing on smartphones. Yang *et al.* [10] proposed to save energy for video streaming by adaptively adjusting the CPU frequency. Geng *et al.* [11] proposed an energy-efficient computational offloading scheme for energy-intensive video processing on

multicore-based mobile devices. He *et al.* [36] proposed to save energy through dynamic resolution scaling based on the user-screen distance. Kim *et al.* [38] adjusted the refresh rate of the screen based on video content to achieve energy saving. Park *et al.* [39] proposed to save energy by applying adaptive frame rate to some parts of the video. Other researchers [12], [13], [37] focused on dynamically adapting the video bitrate and the display brightness to save energy. Complementary to these energy-saving techniques, we take a different approach to save energy by considering the context of video streaming.

7 Conclusions

In this paper, we proposed to save energy by considering the context of video streaming; i.e., watching video on a moving vehicle or in a static environment may have different QoE requirements. Based on quality assessment experiments, we collected traces and modeled the impact of video bitrate and vibration level on QoE, and modeled the impact of video bitrate and signal strength on power consumption. Based on the QoE model and the power model, we formulated the context-aware and energy-aware video streaming problem as an optimization problem. We presented an optimal algorithm which can maximize QoE and minimize energy. Since the optimal algorithm requires perfect knowledge of future tasks which is not available in practice, we then proposed an online bitrate selection algorithm. To further improve the performance of the online algorithm, we proposed a crowdsourcing based approach for bandwidth and vibration prediction. Through real measurements and trace-driven simulations, we demonstrated that our proposed algorithms can significantly reduce the energy consumption (34.8% for the crowdsourcing based algorithm) with only small QoE degradation (3.2% for the crowdsourcing based algorithm).

Although smartphones are used for experiments in this paper, the proposed algorithms can also be applied to other mobile devices, such as tablets, considering their special features. The power model can be enhanced by considering other factors such as the current level of the battery. Besides using the accelerometer sensor, other information such as WiFi SSID can be leveraged for inferring the context of video streaming, at home, in a commuter train, or bus. We will also investigate how to apply the proposed techniques to other video streaming scenarios such as 360 degree video streaming.

Acknowledgments

This work was supported in part by the National Science Foundation under grant CNS-1815465.

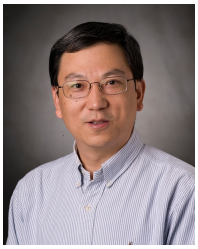
References

[1] Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 2016–2021 White Paper. http://goo.gl/DXWFyr.
[2] J. Jiang, V. Sekar, and H. Zhang. Improving Fairness, Efficiency, and Stability in HTTP-based Adaptive Video Streaming With FESTIVE. *IEEE/ACM Trans. on Networking*, 2014.
[3] A. H. Zahran, D. Raca, and C. Sreenan. ARBITER+: Adaptive Rate-Based Intelligent HTTP Streaming Algorithm for Mobile Networks. *IEEE Trans. on Mobile Computing*, 2018.
[4] Y. Qin, R. Jin, S. Hao, K. R. Pattipati, F. Qian, S. Sen, B. Wang, and C. Yue. A Control Theoretic Approach to ABR Video Streaming: A Fresh Look at PID-based Rate Adaptation. In *IEEE INFOCOM*, 2017.
[5] K. Spiteri, R. Urgaonkar, and R. K. Sitaraman. BOLA: Near-Optimal Bitrate Adaptation for Online Videos. In *IEEE INFOCOM*, 2016.
[6] R. Pérez-Torres, C. Torres-Huitzil, and H. Galeana-Zapién. Power Management Techniques in Smartphone-based Mobility Sensing Systems: A Survey. *Pervasive and Mobile Computing*, 2016.
[7] W. Hu and G. Cao. Energy-Aware Video Streaming on Smartphones. In *IEEE INFOCOM*, 2015.
[8] Y. Yang and G. Cao. Prefetch-Based Energy Optimization on Smartphones. *IEEE Trans. on Wireless Communications*, 2018.
[9] M.A. Hoque, M. Siekkinen, , and J.K. Nurminen. Using Crowd-Sourced Viewing Statistics to Save Energy in Wireless Video Streaming. In *ACM MobiCom*, 2013.
[10] Y. Yang, W. Hu, X. Chen, and G. Cao. Energy-Aware CPU Frequency Scaling for Mobile Video Streaming. *IEEE Trans. on Mobile Computing*, 2019.
[11] Y. Geng, Y. Yang, and G. Cao. Energy-Efficient Computation Offloading for Multicore-Based Mobile Devices. In *IEEE INFOCOM*, 2018.
[12] J. S. Leu, M. C. Yu, C. Y. Liu, A. P. Budiarsa, and V. Utomo. Energy Efficient Streaming for Smartphone by Video Adaptation and Backlight Control. *Computer Networks*, 2017.
[13] Z. Yan and C. W Chen. RnB: Rate and Brightness Adaptation for Rate-Distortion-Energy Tradeoff in HTTP Adaptive Streaming over Mobile Devices. In *ACM MobiCom*, 2016.
[14] X. Sun, Z. Lu, W. Hu, and G. Cao. SymDetector: Detecting Sound-Related Respiratory Symptoms Using Smartphones. In *ACM UbiComp*, 2015.
[15] J. Hao, R. Zimmermann, and H. Ma. GTube: Geo-Predictive Video Streaming over HTTP in Mobile Environments. In *ACM Int'l Conf. on Multimedia Systems*, 2014.
[16] B. Taani and R. Zimmermann. Spatio-Temporal Analysis of Bandwidth Maps for Geo-Predictive Video Streaming in Mobile Environments. In *ACM Int'l Conf. on Multimedia*, 2016.
[17] ITU-T P.910: Subjective Video Quality Assessment Methods for Multimedia Applications. https://www.itu.int/rec/T-REC-P.910-200804-I.
[18] K. Yamagishi and T. Hayashi. Parametric Quality-Estimation Model for Adaptive-Bitrate-Streaming Services. *IEEE Trans. on Multimedia*, 2017.
[19] R. K. Mok, E. W. Chan, and R. K. Chang. Measuring the Quality of Experience of HTTP Video Streaming. In *IFIP/IEEE Symposium on Integrated Network Management (IM)*, 2011.
[20] X. Yin, A. Jindal, V. Sekar, and B. Sinopoli. A Control-Theoretic Approach for Dynamic Adaptive Video Streaming over HTTP. In *ACM SIGCOMM*, 2015.
[21] G. Cermak, M. Pinson, and S. Wolf. The Relationship Among Video Quality, Screen Resolution, and Bit Rate. *IEEE Trans. on Broadcasting*, 2011.
[22] B. Belmudez and S. Moller. An Approach for Modeling the Effects of Video Resolution and Size on the Perceived Visual Quality. In *IEEE Int'l Symposium on Multimedia*, 2011.
[23] Quadratic Function. http://mathworld.wolfram.com/QuadraticCurve.html.
[24] I. Y. Kim and O. L. De Weck. Adaptive Weighted Sum Method for Multiobjective Optimization: a New Method for Pareto Front Generation. *Structural and Multidisciplinary Optimization*, 31(2):05–116, 2006.
[25] Inverse Distance Weighting. https://en.wikipedia.org/wiki/Inverse-distance-weighting.
[26] T. Y. Huang, R. Johari, N. McKeown, M. Trunnell, and M. Watson. A Buffer-based Approach to Rate Adaptation: Evidence from a Large Video Streaming Service. In *ACM SIGCOMM*, 2014.
[27] Y. Liu, S. Dey, F. Ulupinar, M. Luby, and Y. Mao. Deriving and Validating User Experience Model for DASH Video Streaming. *IEEE Trans. on Broadcasting*, 2015.
[28] Z. Li, X. Zhu, J. Gahm, R. Pan, H. Hu, A. C. Begen, , and D. Oran. Probe and Adapt: Rate Adaptation for HTTP Video Streaming at Scale. *IEEE J. Selected Areas in Communications*, 2014.
[29] H. Mao, R. Netravali, and M. Alizadeh. Neural Adaptive Video Streaming with Pensieve. In *ACM SIGCOMM*, 2017.
[30] V. Balasubramanian, M. Wang, M. Reisslein, and C. Xu. Edgeboost: Enhancing Multimedia Delivery with Mobile Edge Caching

in 5G-D2D Networks. In *IEEE Int'l Conf. on Multimedia and Expo (ICME)*, 2019.
[31] W. Jiang, G. Feng, S. Qin, T. Yum, and G. Cao. Multi-agent reinforcement learning for efficient content caching in mobile d2d networks. *IEEE Trans. on Wireless Communications*, 2019.
[32] T. Enghardt, T. Zinner, and A. Feldmann. Using Informed Access Network Selection to Improve HTTP Adaptive Streaming Performance. In *ACM Int'l Conf. on Multimedia Systems*, 2020.
[33] J. Wu, B. Cheng, M. Wang, and J. Chen. Energy-Efficient Bandwidth Aggregation for Delay-Constrained Video over Heterogeneous Wireless Networks. *IEEE J. Selected Areas in Communications*, 2017.
[34] Y. Yang, Y. Geng, and G. Cao. Energy-Aware Advertising Through Quality-Aware Prefetching on Smartphones. In *IEEE MASS*, 2017.
[35] J. Huang, F. Qian, A. Gerber, Z. M. Mao, S. Sen, and O. Spatscheck. A Close Examination of Performance and Power Characteristics of 4G LTE Networks. In *ACM MobiSys*, 2012.
[36] S. He, Y. Liu, and H. Zhou. Optimizing Smartphone Power Consumption through Dynamic Resolution Scaling. In *ACM MobiCom*, 2015.
[37] B. Varghese, G. Jourjon, K. Thilakarathne, and A. Seneviratne. e-DASH: Modelling An Energy-Aware DASH Player. In *IEEE Int'l Symp. on "A World of Wireless, Mobile and Multimedia Networks" (WoWMoM)*, 2017.
[38] D. Kim, N. Jung, Y. Chon, and H. Cha. Content-Centric Energy Management of Mobile Displays. *IEEE Trans. on Mobile Computing*, 2016.
[39] K. Park and M. Kim. EVSO: Environment-Aware Video Streaming Optimization of Power Consumption. In *IEEE INFOCOM*, 2019.

Xianda Chen received the BS degree in software engineering from Northwestern Polytechnical University, China, and the MS degree in electrical and computer engineering from Sungkyunkwan University, South Korea. He is currently pursuing the Ph.D. degree with the Department of Computer Science and Engineering, the Pennsylvania State University, USA. His research interests include wireless networks, mobile computing, and mobile video. He is a student member of the IEEE.

Tianxiang Tan received the BE degree from Sun Yat-sen University and the MS degree in computer science from University of Southern California. He is currently working toward the PhD degree in the Department of Computer Science and Engineering, the Pennsylvania State University. His research interests include mobile cloud computing, edge computing and deep learning. He is a student member of the IEEE.



Guohong Cao received his B.S. degree in computer science from Xi'an Jiaotong University, and his Ph.D. in computer science from the Ohio State University in 1999. Since then, he has been with the Department of Computer Science and Engineering at the Pennsylvania State University, where he is currently a Distinguished Professor. He has published more than 200 papers in the areas of wireless networks, mobile computing, machine learning, wireless security and privacy, and Internet of Things, which have been cited over 20000 times. He has served on the editorial board of IEEE Transactions on Mobile Computing, IEEE Transactions on Wireless Communications, and IEEE Transactions on Vehicular Technology, and has served on the organizing and technical program committees of many conferences, including the TPC Chair/Co-Chair of IEEE SRDS, MASS, and INFOCOM. He has received several best paper awards, the IEEE INFOCOM Test of Time award, and the NSF CAREER award. He is a Fellow of the AAAS and a Fellow of the IEEE.

Thomas F. La Porta is the Director of the School of Electrical Engineering and Computer Science and Penn State University. He is an Evan Pugh Professor and the William E. Leonhard Chair Professor in the Computer Science and Engineering Department and the Electrical Engineering Department. He received his B.S.E.E. and M.S.E.E. degrees from The Cooper Union, New York, NY, and his Ph.D. degree in Electrical Engineering from Columbia University, New York, NY. He joined Penn State in 2002. He was the founding Director of the Institute of Networking and Security Research at Penn State. Prior to joining Penn State, Dr. La Porta was with Bell Laboratories for 17 years. He was the Director of the Mobile Networking Research Department in Bell Laboratories, Lucent Technologies where he led various projects in wireless and mobile networking. He is an IEEE Fellow, Bell Labs Fellow, received the Bell Labs Distinguished Technical Staff Award, and an Eta Kappa Nu Outstanding Young Electrical Engineer Award. He also won two Thomas Alva Edison Patent Awards. His research interests include mobility management, signaling and control for wireless networks, security for wireless systems, mobile data systems, and protocol design. Dr. La Porta was the founding Editor-in-Chief of the IEEE Transactions on Mobile Computing. He has published numerous papers and holds 39 patents.