REQUEST: Seamless Dynamic Adaptive Streaming over HTTP for Multi-Homed Smartphone under Resource Constraints

Jonghoe Koo
Department of ECE and INMC,
Seoul National University
Seoul, Korea
jhkoo@mwnl.snu.ac.kr

Juheon Yi
Department of ECE and INMC,
Seoul National University
Seoul, Korea
jhyi16@mwnl.snu.ac.kr

Joongheon Kim
School of Computer Science and Engineering, Chung-Ang University
Seoul, Korea
joongheon@cau.ac.kr

Mohammad Ashraful Hoque
University of Helsinki
Helsinki, Finland
mohammad.a.hoque@helsinki.fi

Sunghyun Choi
Department of ECE and INMC,
Seoul National University
Seoul, Korea
schoi@snu.ac.kr

ABSTRACT

Exploiting both LTE and Wi-Fi links simultaneously enhances the performance of video streaming services in a smartphone. However, it is challenging to achieve seamless and high quality video while saving battery energy and LTE data usage to prolong the usage time of a smartphone. In this paper, we propose REQUEST, a video chunk request policy for Dynamic Adaptive Streaming over HTTP (DASH) in a smartphone, which can utilize both LTE and Wi-Fi. REQUEST enables seamless DASH video streaming with near optimal video quality under given budgets of battery energy and LTE data usage. Through extensive simulation and measurement in a real environment, we demonstrate that REQUEST significantly outperforms other existing schemes in terms of average video bitrate, rebuffering, and resource waste.

KEYWORDS

DASH, video streaming, LTE, Wi-Fi, smartphone, energy, LTE data quota, stochastic optimization

1 INTRODUCTION

Multi-homing enables smartphones to utilize multiple network interfaces, e.g., LTE and Wi-Fi, simultaneously to enhance the quality of experience (QoE) for the end users. Among various mobile applications, dynamic adaptive streaming over HTTP (DASH) is a very good candidate that can take advantage of multi-homing towards better QoE [7, 10]. Mobile users can enjoy high-quality video over Wi-Fi in high-bandwidth Wi-Fi hotstpots, and watch video contents over LTE whenever LTE base stations can support data communication even though there are no available Wi-Fi access points (APs). In a place where both LTE and Wi-Fi are available, higher quality videos can be supported or more resource efficient streaming becomes possible by judicially using both LTE and Wi-Fi links simultaneously.

Let us consider a video streaming scenario, where user wants to enjoy content with long playback duration, e.g., a live soccer game or video on demand (VoD) contents from YouTube or Netflix. Since the content duration is very long, Wi-Fi may be the desirable access network for the user. However, the user might be on move or take public transport while experiencing the content. Such mobility results in extremely fluctuating link throughput or frequently disconnected Wi-Fi links from the nearby hotspots [6, 17]. This significantly degrades video quality and causes frequent rebuffering (also called video stall or freezing), and hence, in this environment, it is not desirable to watch video over Wi-Fi link.

To tackle this problem, an intelligent video chunk requesting mechanism should be developed that can utilize the unstable Wi-Fi links intelligently while guaranteeing the desired video quality and uninterrupted playback without rebuffering in mobile environments. At the same time, battery energy and LTE data quota are important issues which are desired to be conserved as much as possible while ensuring satisfying video quality.

Accordingly, a chunk request technique should consider both enhancing video performances, i.e., video quality and rebuffering, and saving smartphone resources, i.e., battery energy and LTE data quota, to optimize QoE of video streaming. Besides, the problem is more complex in a multi-homing environment compared with a single link scenario [11], thus making it challenging to develop a well-designed DASH streaming client, and a solution is still due.

To provide an optimal video performance while satisfying resource usage constraints for battery energy and LTE data quota, we propose REQUEST, a bit*R*ate, *E*nergy, LTE data *Q*uota, and b*U*ff*E*r-aware video *ST*reaming, for DASH video over multi-homed smartphones. By utilizing Wi-Fi link as a supplementary link, REQUEST realizes a seamless DASH video streaming even in the environments where Wi-Fi link performance is not guaranteed. Also, REQUEST optimizes time-average video quality while satisfying time-average resource constraints by adopting Lypunov optimization framework, and it is easily implemented in commercial smartphones as an application. We claim the following major contributions:

MM '17, October 23–27, 2017, Mountain View, CA, USA
© 2017 Association for Computing Machinery.
ACM ISBN 978-1-4503-4906-2/17/10…$15.00
https://doi.org/10.1145/3123266.3123368

- We propose a chunk request policy that achieves seamless playback of video using both Wi-Fi and LTE simultaneously even in situations where Wi-Fi is unstable.
- We formulate a Lyapunov optimization framework-based stochastic optimization problem to maximize time-average video quality under time-average energy and LTE data usage constraints and minimize rebuffering.
- We design REQUEST, an online video chunk request algorithm by using both LTE and Wi-Fi links, which provides a near-optimal solution of the Lyapunov optimization problem.
- We implement REQUEST by modifying ExoPlayer, Google's open-source DASH player for Android [3], and validate its performance in real-world scenarios.

The rest of the paper is organized as follows. In Section 2, we discuss issues arising when utilizing both LTE and Wi-Fi for DASH video streaming and related work. Section 3 describes motivation of our work, and our proposed chunk request policy is presented in Section 4. We formulate the optimization problem in Section 5 and REQUEST algorithm is introduced in Section 6. We evaluate the performance of REQUEST in Section 7 and conclude the paper in Section 8.

2 BACKGROUND AND RELATED WORK

2.1 Background

2.1.1 Multi-homing for DASH. DASH is a bitrate-adaptive streaming technique, which delivers video content over HTTP. Video content is encoded at a variety of bitrates in video server. A DASH-enabled video streaming player downloads video chunks of a particular bitrate based on the experienced bandwidth for downloading earlier chunks. Initially, the player begins with lower or moderate quality to avoid longer start-up delay. Bandwidth aggregation in heterogeneous wireless network environment aims to support higher bitrates of video that cannot be sufficiently delivered by only one of the available networks [30–33, 35]. Thanks to the development of techniques for simultaneous utilization of multiple network interfaces at mobile devices, especially, as application-level solutions [2, 4], both Wi-Fi and LTE interfaces can be utilized simultaneously to enhance video quality. The higher the video quality, the higher the energy consumption of smartphones, because the amount of data to be decoded and downloaded via Wi-Fi or LTE increases [10, 16]. The energy is also spent by the CPU for processing more packets [16]. Meanwhile, user may not fully utilize LTE link due to her/his remaining monthly LTE data quota or data plan. In this case, it is fairly useful to download video data by Wi-Fi as much as possible to save LTE data quota [9, 17].

By downloading video chunks with both LTE and Wi-Fi, video player can maintain sufficient chunks in its buffer, thus avoiding rebuffering. However, chunks remaining in the buffer may cause possible data waste when user stops watching video in the middle of playback [11, 18]. Prefetching video data, i.e., requesting more video chunks in advance, provides several advantages to video client. The more video chunks to be consecutively requested, the less energy is spent for networking activities as network interfaces are able to stay in idle state for longer periods [11, 13, 18]. In addition, prefetching prevents rebuffering events because video buffer is filled with sufficiently many video chunks. However, large amount of prefetching may waste as much energy and LTE data as the number of chunks remaining in the video buffer when user stops watching video before the end of the video clip. The ratio of wasted energy and LTE data by prefetching large amount of chunks becomes significant, especially when user abandons video streaming session quite early. Therefore, it is necessary to determine an optimal prefetching strategy considering the advantages of prefetching (prevention of rebuffering and energy efficiency) and disadvantages (waste of energy and LTE data due to user's leaving) to enhance QoE of video consumers.

2.1.2 Lyapunov optimization framework. Stochastic optimization aims at minimizing a time-average objective function subject to queue stability when the utility function and queue stability conditions are in tradeoff relationship. In addition, the stochastic optimization framework is able to utilize the concept of virtual queues for time-average constraint representation. In stochastic optimization formulation, Lyapunov function $L(t)$ is defined as the sum of squares of backlogs in actual and virtual queues on a slot (in a slotted system). Lyapunov drift $\Delta(t)$ is defined as the difference in the Lyapunov function per slot time. While pursuing the minimization of a time-average objective function, taking the minimum of the Lyapunov drift leads to the queue stability (i.e., main constraint), which is referred to as *drift-plus-penalty* minimization. By taking an action to greedily minimize the drift-plus-penalty every slot time, we can achieve a time-average utility deviating by at most $O(1/V)$ from optimality while satisfying time-average constraints and a time-average queue backlog bound of $O(V)$, where V is defined as a tradeoff factor between utility and queue stability. For further details about the theory of Lyapunov optimization, the book [21] can be referred to.

2.2 Related Work

Energy/cellular data quota-aware video streaming: Previous efforts [11, 19] control a video segment size to be prefetched in an HTTP-based video streaming service to improve the energy efficiency. *GreenTube* [19] aims to minimize an unnecessary active period of 3G/4G radios by scheduling each video segment downloading. *eSchedule* [11] utilizes crowd-sourced video viewing statistics and power models for energy efficient scheduling of video streaming. *QAVA* [8] manages the tradeoff between cellular data usage and video quality by predicting video client's usage behavior. *QAVA* automatically selects optimal video quality to enable users to keep under their data quota while maximizing video quality. Lee *et al.* [18] and Hu *et al.* [12] consider wastage of energy and cellular data quota incurring when user stops watching video. Both consider the scenario where video streaming is conducted only via LTE interface of a smartphone. *GreenBag* [7] utilizes both LTE and Wi-Fi links to achieve better quality of service (QoS) and energy efficiency. Go *et al.* [10] propose an energy-aware HTTP adaptive video streaming under a cellular data usage constraint. It considers simultaneous usage of LTE and Wi-Fi for DASH video streaming on smartphones. It selects video bitrate and the number of chunks to request via each network in order to minimize a weighted sum of video distortion and energy consumption per video chunk.

Optimal bitrate selection of DASH video: Video client selects a video chunk with an appropriate bitrate according to current network status [5, 27] or video player's buffer status [14]. Due to severe fluctuation of link throughput in mobile environment, it is difficult for a link throughput-based bitrate adaptation to practically function, and for this reason, a buffer-based bitrate adaptation has been studied and practically used by real implementations [14, 29].

3 MOTIVATION

3.1 Wi-Fi Throughput Fluctuation

While a mobile device can utilize both Wi-Fi and LTE networks simultaneously, the Wi-Fi link may become unstable and its quality may fluctuate severely especially in a mobile or dense environment. For example, when a user moves around and goes out of the coverage of a Wi-Fi AP connected to the user's mobile device, the device cannot utilize Wi-Fi link until it finds an available Wi-Fi AP nearby and handover to a new Wi-Fi AP is successfully completed. It is also well known that Wi-Fi throughput degradation occurs in mobile environments due to poor performance of handover operations in commercial Wi-Fi devices [20, 24, 26]. In addition, if a Wi-Fi AP operates at 2.4 GHz, the Wi-Fi link quality may severely suffer from interference caused by other wireless devices operating in 2.4 GHz ISM band, such as Bluetooth, ZigBee, microwave ovens, and cordless phones [15, 25]. Furthermore, Wi-Fi at 5 GHz suffers from higher path loss than Wi-Fi at 2.4 GHz, thus increasing the possibility that mobile user experiences worse Wi-Fi link quality and goes out of Wi-Fi coverage. In contrast to Wi-Fi, a device may retain a seamless connection via the LTE network even though user moves fast, thanks to seamless handover between LTE base stations.[1] Likewise, in mobile and dense environments, Wi-Fi link may have more unstable quality and sometimes may be unavailable.

However, Wi-Fi can be still utilized for offloading purpose even in mobile environment [6, 17], and its offloading capability may increase in static environment or when a device associates to an IEEE 802.11ac-compliant Wi-Fi AP that can provide very high throughput. From this perspective, when a DASH client requests video chunks via both Wi-Fi and LTE in parallel in a multi-homed device, Wi-Fi link availability is like a double-edged sword. In other words, although the Wi-Fi link may enhance the video quality at low energy cost and reduce LTE data consumption, it can also increase the possibility of rebuffering events, as we cannot predict the exact bandwidth and availability of Wi-Fi in mobile and dense environments.

Therefore, it is challenging to design a chunk request policy for DASH by judicially utilizing Wi-Fi connectivity in addition to LTE to maximize the merit of utilizing Wi-Fi link while minimizing its side effects. In this work, we accept the challenge. We opportunistically request video chunks over Wi-Fi, thus reducing the side effects of unstable Wi-Fi links.

3.2 Optimizing Resource Utilization

Mobile devices consume resources, i.e., battery energy and LTE data quota, during DASH video streaming over Wi-Fi and LTE networks. Since battery energy and remaining LTE data quota are usually limited, users will like to minimize the resource usage for DASH video streaming as much as possible so as to watch videos much longer or use the remaining resources for other applications. Assuming that a DASH client intelligently requests proper bitrate video chunks to balance the quality of video and resource usage. In this case, the video quality might be either increased in exchange of using more resources or decreased to conserve the resources. From this perspective, a problem can be formulated as optimizing video quality for DASH streaming given the constrained amount of resources in smartphones, i.e., requesting high quality video chunks with a given amount of battery energy and LTE data quota.

Unfortunately, the traditional optimization frameworks that have been used in the past studies do not support flexible resource utilization, , e.g., sometimes allowing resource overuse to enhance video quality. For instance, a popular method to formulate optimization problem is to formulate a multi-attribute cost function with proper constraints based on a simple additive weighting (SAW) method. This approach is widely used for multi-attribute decision making (MADM) algorithms [10, 23, 28]. A challenge to optimize MADM-based cost function is to select appropriate weights for the attributes in the cost function to quantitatively balance them. In addition, even though the weights of the attributes determine relative priority to provide a trade-off between attributes in a cost function, it is difficult to force an attribute to have a value within a certain range.

To overcome the limitation of MADM-based optimization, we adopt Lyapunov optimization framework to optimize time-average video quality while satisfying time-average resource constraints at the same time. By adopting time-average concepts, we occasionally allow resource overuse but eventually satisfy constraints.

4 PROPOSED CHUNK REQUEST POLICY

Generally, before a DASH client sends a request for a video chunk, it determines an appropriate bitrate for the chunk according to the estimated link bandwidth [10] or the playback buffer status [14, 29]. In a multi-homed environment, a DASH client can further download a single chunk in parallel via multiple wireless interfaces by sending multiple HTTP range requests [1]. A number of proposed approaches follow this technique [7, 10]. However, considering a possible situation where Wi-Fi link is disconnected or extremely unstable during a chunk download, it is not efficient to divide one video chunk into two parts and receive them by both LTE and Wi-Fi separately. Accordingly, in this work, we use a single network to request and receive a single video chunk. This eliminates decoding failures due to partially received chunks, thus avoiding re-buffering and data wastage.

Therefore, we design our chunk request policy as illustrated in Fig. 1, where the notations in Fig. 1 are summarized in Table 1. We call an event of requesting a batch of video chunks a *request event*. A *request interval* $T[r]$ is defined as the time interval between the start time and the end time of a *request event*. At the start time $t[r]$ of the rth *request event*, client determines the bitrate $b[r]$ of chunks,[2] *request interval* $T[r]$, and the numbers of chunks to request over LTE ($N_l[r]$) and Wi-Fi ($N_w[r]$) during $T[r]$. Ideally, all the chunks requested during $T[r]$, i.e., $N_l[r] + N_w[r]$ chunks, are expected to be successfully downloaded within $T[r]$. In this case, $t[r+1] = t[r] + T[r]$, i.e., the end time of the rth *request event* is equal to the start time of the $(r+1)$th *request event*.

[1] In this paper, it is assumed that seamless handover of LTE links is ensured. Other cellular links, e.g., 3G/HSDPA, may be applied instead of LTE.

[2] All the chunks requested in a *request event* have the same bitrate.

Table 1: Important notations

Symbol	Description
t_p	A fixed chunk playback time (sec)
T_b	Initial buffer-time (sec)
r	The index of request event
$t[r]$	Start time of the rth request event
$T[r]$	The rth request interval ($t[r+1] - t[r]$)
$b[r]$	Bitrate of chunks to be requested in the rth request event
$N_l[r]$	Number of chunks to request over LTE in the rth request event
$N_w[r]$	Number of chunks to request over Wi-Fi in the rth request event
$\hat{N}_w[r]$	Number of chunks actually received over Wi-Fi in the rth request event

Moreover, it is important to determine $T[r]$ judiciously to ensure that all the chunks are downloaded before they are played back for seamless video streaming. Especially, our policy tries to avoid rebuffering due to Wi-Fi throughput degradation. To enable this goal, the following features are introduced in our design:

i) We first set $T[r] = t_p(\hat{N}_w[r-1] + N_l[r])$, i.e., the playback time of $\hat{N}_w[r-1]$ chunks received over Wi-Fi in the $(r-1)$th *request event* and $N_l[r]$ chunks requested over LTE in the rth *request event*. By doing so, the chunks received over Wi-Fi accumulate in the buffer, which has the same effect as increasing the initial buffer-time at the start of the next *request event*. It also prevents rebuffering due to LTE throughput degradation that cannot be handled by the initial buffer.

ii) The video chunks requested over Wi-Fi in the rth *request event* should be played after T_b from the end time of the rth *request event*, where T_b is initial buffer-time.[3] For example, if $T_b = 2t_p$, $\hat{N}_w[r]$ chunks received over Wi-Fi in the rth *request event* are played after $t[r] + T[r] + 2t_p$ as illustrated in Fig. 1.

iii) If chunks are requested over both LTE and Wi-Fi, the video chunk requested over Wi-Fi is played right after the last video chunk requested over LTE in the same *request event*. In other words, the first sequence of $N_w[r]$ chunks is the right next to the last sequence of $N_l[r]$.

iv) If client fails to request some chunks over Wi-Fi, those chunks are prioritized to request in the next *request event*. For example, as illustrated in Fig. 1, if two chunks could not be requested in the rth *request event*, the client starts requesting the chunks in order in the $(r+1)$th *request event* for the continuity of video.

Based on these features, we ensure that all chunks of a video can be eventually received and played even though some chunks could not be requested over Wi-Fi during a *request event*. However, in practice, chunks may not be received within a *request interval*, due to throughput degradation of either LTE or Wi-Fi. In this case, the start of the next *request event* can be delayed. The details are discussed in Section 6.1. In addition, to maximize video quality while satisfying battery energy and LTE data quota constraints, we should properly determine b, T, N_l, and N_w for each *request event*. To achieve this, we formulate a stochastic optimization problem in the following section.

5 PROBLEM FORMULATION

Even if Wi-Fi and LTE links are sufficient to support the highest video quality, we cannot request high quality video if the battery energy and/or LTE data usage is limited. Thus, we have to accurately determine the bitrate of chunks and the number of requested chunks

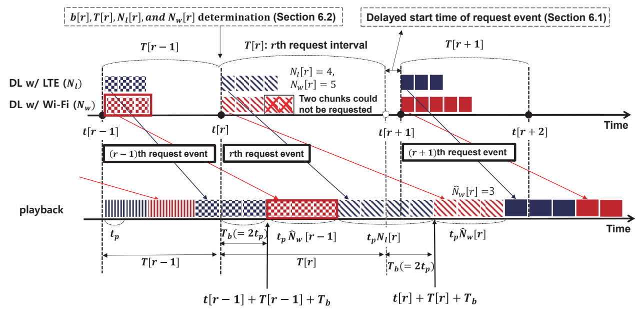


Figure 1: An example of the proposed chunk request policy.

to maximize video quality while satisfying battery and LTE data usage constraints during the entire video playback.

Objective: The objective of our optimization problem is to maximize a time-average video utility. In this work, we define the time-average utility as a time-average logarithmic function of the video bitrate. The reason why we maximize the logarithmic function of bitrate instead of bitrate directly is to reflect the fact that the impact of increasing video quality on a user experience can be modeled by a concave function with diminishing returns [29].

Constraints: To guarantee longer battery lifetime and use LTE data without exceeding budget, we assume that mobile device is allowed to consume p_{av} (W) and consume LTE data with d_{av} (Mbps) on average during a video playback.

In order to design optimal chunk request policy for maximizing the time-average video quality with satisfying time-average resource constraints, it would require *a priori* statistical knowledge, such as the distribution of network bandwidth. It would also need a complex computational method, such as dynamic programming (DP) method for finding the optimal solution based on the *a priori* statistical knowledge [34]. However, it is practically difficult to obtain the exact distribution of network bandwidth to solve the problem with a DP method. Even if the distribution could be obtained, very large state space composed of request time, bitrate, number of chunks to request via network would have to be constructed [22, 29].

In order to overcome this challenge, we apply Lyapunov optimization-based dynamic algorithm [22] that independently determines the number of chunks to request and their bitrate at the start of a *request event* by observing the chunk reception result of the previous *request event*. The performance of Lyapunov-based dynamic algorithm is close to that of the optimal solution by a DP algorithm with *a priori* statistical knowledge, but Lyapunov optimization algorithm does not require any *a priori* knowledge [22].

Renewal-based Lyapunov optimization: According to the chunk request policy proposed in Section 4, *request interval* $T[r]$ is related to the number of requested chunks, and hence, it is time-varying. To reflect this, we adopt renewal-based Lyapunov optimization [21, 22].[4] We assume that the video playback time is infinite, i.e., $R \longrightarrow \infty$,[5] for an approach similar to that in [29]. Then, the

[3]The initial buffer-time is the playback time of video chunks initially received before video player starts rendering the video.

[4]A *request event* in Fig. 1 corresponds to a renewal frame in the renewal-based Lyapunov optimization.

[5]R is the index of the last *request event* for a video.

problem based on Lyapunov optimization framework to maximize time-average video utility while satisfying time-average energy and LTE data usage constraints is described as follows:

$$\begin{aligned} \text{Maximize} \quad & \overline{\log(b)T}/\overline{T}, \\ \text{subject to} \quad & \overline{e}_e \le p_{av}\overline{T}, \;\; \overline{e}_d \le d_{av}\overline{T}, \end{aligned} \tag{1}$$

where $\overline{\log(b)T}$, $\overline{T}$, $\overline{e}_e$, and $\overline{e}_d$ are infinite horizon expectations, i.e.,

$$(\overline{\log(b)T}, \overline{T}, \overline{e}_e, \overline{e}_d) = \lim_{R\to\infty} \frac{1}{R} \sum_{r=0}^{R-1} (\log(b[r])T[r], T[r], e_e[r], e_d[r]), \tag{2}$$

where $e_e[r]$ and $e_d[r]$ are the energy consumption and LTE data usage during the rth *request event*, respectively. Since *request interval* is time-varying, we give a time weight to $\log(b)$, thus maximizing $\overline{\log(b)T}/\overline{T}$, which denotes the time-average video utility.

Virtual queues for resource constraints: We define virtual queues to solve our optimization problem based on a drift-plus-penalty ratio minimization algorithm, which greedily minimizes drift-plus-penalty ratio to achieve near-optimal time-average video utility while satisfying time-average resource constraints.

Let us first define a virtual queue for energy consumption $Q_e[r]$. The arrival and service process of $Q_e[r]$ at the rth *request event* are $e_e[r]$ and $p_{av}T[r]$, respectively. Then, the queue backlog of $Q_e[r]$ evolves as follows:

$$Q_e[r+1] = \max\left(Q_e[r] + e_e[r] - p_{av}T[r], 0\right). \tag{3}$$

The same approach can be applied to the LTE data usage virtual queue $Q_d[r]$ of which backlog evolves as follows:

$$Q_d[r+1] = \max\left(Q_d[r] + e_d[r] - d_{av}T[r], 0\right). \tag{4}$$

Time-average resource constraints of (1) can be satisfied if $Q_e[r]$ and $Q_d[r]$ are stabilized.[6]

Drift-plus-penalty ratio: The renewal-based Lyapunov optimization framework minimizes drift-plus-penalty ratio which is obtained by normalizing drift-plus-penalty by time [22]. Before defining the drift-plus-penalty ratio, we first define the Lyapunov function $L[r]$. In addition to Q_e and Q_d, we consider DASH client's video buffer $Q_r[r]$ which denotes the number of video chunks in video buffer at the rth *request event*. Then, $L[r]$ is defined by:

$$L[r] = \frac{1}{2}\left((Q_e[r])^2 + (Q_d[r])^2 + \left(\frac{Q_{\max}}{2} - Q_r[r]\right)^2 \right), \tag{5}$$

where $Q_{\max}$ is defined as the maximum number of chunks stored in the video buffer. The energy consumption and LTE data usage queues are desired to be empty to satisfy the time-average constraints while the video chunk queue should not be empty to avoid rebuffering. However, filling the video buffer to Q_{max} is not desirable due to the possibility of queue overflow and waste of resources, and hence, we use $\frac{Q_{max}}{2}$ instead of Q_{max} when defining $L[r]$.

Then, the Lyapunov drift $\Delta[r]$, which is desired to be minimized, is defined as the difference between the Lyapunov function of the rth *request event* and that of the $(r+1)$th *request event*.

$$\Delta[r] = L[r+1] - L[r]. \tag{6}$$

Conventional Lyapunov optimization framework minimizes penalty. Since our work maximizes video utility, we define penalty as $-\log(b)T$ to follow the Lyapunov framework's concept of penalty minimization. Finally, the drift-plus-penalty ratio (DPP_r) is defined as:

$$DPP_r[r] = \frac{\Delta[r] + V \cdot penalty[r]}{T[r]} = \frac{\Delta[r]}{T[r]} - V\log(b[r]), \tag{7}$$

where V is a positive constant parameter to allow a tradeoff between virtual queues' backlogs and the gap from a theoretical optimal video utility. By taking an action to greedily minimize the drift-plus-penalty ratio every *request event*, we can achieve time-average video utility maximization deviating by at most $O(1/V)$ from optimality while satisfying time-average resource constraint bound of $O(V)$.

[6]Due to the lack of space, we omit the proof.

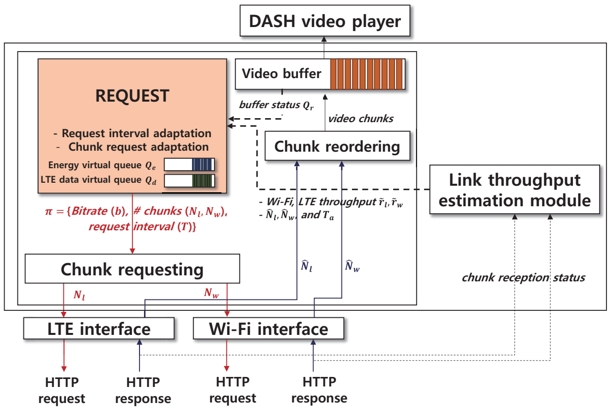

DASH video player
REQUEST
- Request interval adaptation
- Chunk request adaptation
Energy virtual queue Q_e
LTE data virtual queue Q_d
Video buffer
buffer status Q_r
video chunks
Chunk reordering
Link throughput estimation module
$\pi = \{$Bitrate (b), # chunks (N_l, N_w), request interval $(T)\}$
- Wi-Fi, LTE throughput $\tilde{r}_l, \tilde{r}_w$
- $\hat{N}_l, \hat{N}_w$, and T_a
Chunk requesting
$\hat{N}_l$
$\hat{N}_w$
chunk reception status
N_l
N_w
LTE interface
Wi-Fi interface
HTTP request
HTTP response
HTTP request
HTTP response

Figure 2: REQUEST architecture.

6 REQUEST ALGORITHM

In this section, we introduce REQUEST, an online algorithm to request video chunks with near-optimal video quality while satisfying resource constraints and preventing video rebuffering. The REQUEST architecture is illustrated as Fig. 2. At the start of every *request event*, REQUEST observes current virtual queue backlogs ($Q_e[r]$ and $Q_d[r]$), video queue backlog ($Q_r[r]$), and estimated LTE and Wi-Fi link throughput ($\tilde{r}_l[r]$ and $\tilde{r}_w[r]$). By using these values, REQUEST determines the four parameters, i.e., video bitrate ($b[r]$), *request interval* ($T[r]$), and the number of chunks to request through LTE and Wi-Fi ($N_l[r]$ and $N_w[r]$), which minimize the drift-plus-penalty ratio, $DPP_r[r]$, in (7). Let the four-tuple ($N_l[r]$, $N_w[r]$, $b[r]$, $T[r]$) be the *request policy* $\pi[r]$ for the rth *request event*.

The main algorithms of REQUEST include *request interval adaptation* and *chunk request adaptation*. The *request interval adaptation* controls *request interval* according to chunk reception status, and the *chunk request adaptation* determines π based on DPP_r minimization algorithm.

6.1 Request Interval Adaptation

Delayed start time of request event: If all the chunks requested in the rth *request event* are successfully downloaded within $T[r]$, $t[r+1]$ is equal to $t[r] + T[r]$. However, in practice, some chunks requested via either LTE or Wi-Fi may not be successfully received on time, i.e., within $T[r]$, due to severe throughput fluctuation. In such a case, we should delay the start time of the next *request event* for successful chunk receptions. If we request $N_l[r]$ and $N_w[r]$

chunks by LTE and Wi-Fi at the rth *request event*, respectively, the delay is allowed until the following conditions are satisfied.

i) We wait until all the chunks requested over LTE ($N_l[r]$) are successfully received.
ii) If a chunk is requested over Wi-Fi before $t[r] + T[r]$, we wait until $t[r] + T[r]$ for full reception of the chunk.

Let the number of actually received chunks during the rth *request event* by LTE and Wi-Fi be $\hat{N}_l[r]$ and $\hat{N}_w[r]$. With the above two conditions, $\hat{N}_l[r] = N_l[r]$ and $\hat{N}_w[r] \leq N_w[r]$. The reason why we do not guarantee the reception of all the chunks requested over Wi-Fi is that we quickly start a new *request event* and adapt the bitrate and the number of chunks for the next *request event* to resolve link throughput degradation.

If the chunk download of the rth *request event* is not completed within $T[r]$ and it takes additional $T_a[r]$ time to finish the remaining chunk reception, the start time of the $(r+1)$th *request event* $t[r+1]$ becomes $t[r] + T[r] + T_a[r]$.

Request interval adaptation: When a delay of T_a occurs, the delay is compensated by reducing the next *request interval* by T_a. If $\hat{N}_w[r-1]$ chunks are received over Wi-Fi and the delay of $T_a[r-1]$ occurs, $T[r]$ is reduced by $T_a[r-1]$ as follows:

$$T[r] = t_p(\hat{N}_w[r-1] + N_l[r]) - T_a[r-1]. \tag{8}$$

By doing so, we can compensate for the amount of buffer that has been reduced due to the delay. For example, even though the rth *request event* is delayed by $T_a[r-1]$, we can ensure that there are $\hat{N}_w[r] + T_b/t_p$ chunks in the buffer at $t[r+1]$ by reducing $T[r]$ by $T_a[r-1]$. Without this adaptation, the delays accumulate and initial buffer-time may not be guaranteed at the start time of a *request event*, thus causing rebuffering. According to our request interval adaptation, the video buffer backlog evolves as:

$$Q_r[r+1] = Q_r[r] + (\hat{N}_w[r] + N_l[r]) - (T[r] + T_a[r])/t_p. \tag{9}$$

6.2 Chunk Request Adaptation

Estimated LTE and Wi-Fi throughput: For the rth request event, the estimated LTE/Wi-Fi throughput, $\tilde{r}_l[r]$ and $\tilde{r}_w[r]$, are calculated by using exponential weighted moving average (EWMA).

$$\tilde{r}_l[r] = \alpha \tilde{r}_l[r-1] + (1-\alpha) r_l[r-1], \tag{10}$$

where $r_l[r-1] = \frac{N_l[r-1] t_p b[r-1]}{\tau_l[r-1]}$, i.e., the average chunk download speed by LTE during the $(r-1)$th *request event*, and $\tau_l[r-1]$ is the total download time for $N_l[r-1]$ chunks of bitrate $b[r-1]$ by the LTE interface. $\tilde{r}_w[r]$ is calculated in a similar fashion.

Arrival rate of energy/LTE data virtual queues: In order to reduce energy and LTE data waste when a user stops watching video in the early stage of playback, we put the expected energy/LTE data waste as well as the expected energy/LTE data consumption in the arrival process of the energy/LTE data virtual queue in the stage of determining the optimal $\pi[r]$. We define $\bar{E}_w[r]$ and $\bar{D}_w[r]$ as the expected energy and LTE data waste during the rth *request event*, respectively.[7]

[7]When updating $Q_e[r]$ and $Q_d[r]$ at $t[r]$, since the video has been played back continuously, set $\bar{E}_w[r-1]$ and $\bar{D}_w[r-1]$ to 0, and then update $Q_e[r]$ and $Q_d[r]$.

The arrival rate $e_e[r]$ of energy virtual queue is given as

$$e_e[r] = e_v[r] + e_{e,w}[r] + e_{e,l}[r] + \bar{E}_w[r] \cdot T[r]/(t[r] + T[r]), \tag{11}$$

where $e_v[r]$ is the energy consumption for video playback including CPU and display power, $e_{e,w}[r]$ and $e_{e,l}[r]$ are the energy consumption for Wi-Fi and LTE interfaces, respectively. We multiply the expected waste by $\frac{T[r]}{t[r]+T[r]}$ to reflect the effect of waste amount on the average power from the beginning of video playback until the end of the current *request event*. The expected energy consumption for network interface n_i, where $n_1 = l$ and $n_2 = w$ for LTE and Wi-Fi, respectively, is calculated as follows:

$$e_{e,n_i}[r] = \tilde{P}_{n_i}[r] t_{n_i,d}[r] + P_{n_i,tail} \cdot \min\left((T[r] - t_{n_i,d}[r]), t_{n_i,tail}\right), \tag{12}$$

where $\tilde{P}_{n_i}[r]$ is the average power for network n_i with the expected throughput $\tilde{r}_{n_i}[r]$, and $t_{n_i,d}[r]$ $\left(= \frac{t_p b[r] N_{n_i}[r]}{\tilde{r}_{n_i}[r]}\right)$ is the expected download time for $N_{n_i}[r]$ chunks of bitrate $b[r]$ by network n_i with the expected throughput $\tilde{r}_{n_i}[r]$. $\tilde{P}_{n_i}[r] t_{n_i,d}[r]$ is the energy consumption of network interface n_i for chunk download and the remaining term denotes the energy consumption for tail time [16].

Likewise, the arrival rate $e_d[r]$ of LTE data virtual queue is

$$e_d[r] = t_p b[r] N_l[r] + \bar{D}_w[r] \cdot T[r]/(t[r] + T[r]). \tag{13}$$

For $\bar{E}_w[r]$ and $\bar{D}_w[r]$, we assume that the probability that a user stops watching video during $T[r]$ takes a uniform distribution. During $T[r]$, the video chunk remaining in the buffer, which are received via Wi-Fi in the previous *request event*, will be played back for $t_p Q_r[r]$, and the chunk received by LTE in the current *request event* will be played back for the remaining time, i.e., $T[r] - t_p Q_r[r]$. We adopt the approach in [18] to derive E_w and D_w as the following closed forms.

$$\begin{aligned}\bar{E}_w[r] = &\frac{1}{T[r]} \left\{ \frac{1}{2}(t_p Q_r[r])^2 e_{e,w}[r-1] + \left(T[r] - \frac{1}{2} t_{l,d}\right) e_{e,l}[r] \right. \\ &\left. + \left(T[r] - \frac{1}{2} t_{w,d}\right) e_{e,w}[r] + \frac{(T[r] - t_p Q_r[r])^2}{2 t_p N_l[r]} e_{e,l}[r] \right\}.\end{aligned} \tag{14}$$

$$\bar{D}_w[r] = \left(1 - \frac{t_{l,d} + t_p N_l[r]}{2T[r]}\right) e_d[r]. \tag{15}$$

Chunk request policy determination algorithm: Now, we find the optimal $\pi[r]$ which minimizes $DPP_r[r]$ as follows:

$$\begin{aligned} \min_{\pi[r]} \quad & DPP_r[r], \\ \text{subject to} \quad & T[r] = t_p\left(\hat{N}_w[r-1] + N_l[r]\right) - T_a[r-1], \\ & \frac{t_p b[r] N_l[r]}{T[r]} \leq \tilde{r}_l[r], \ \frac{t_p b[r] N_w[r]}{T[r]} \leq \tilde{r}_w[r], \\ & N_l[r], N_w[r] \in \{0, 1, 2 \cdots, Q_{max}\}, \\ & 1 \leq N_l[r] + N_w[r] \leq Q_{max}. \end{aligned} \tag{16}$$

The first constraint of (16) is to follow the chunk request policy in Section 4. The second constraint is to prevent requesting more chunks than what the estimated link throughput permits. The third and fourth constraints represent that the video should be requested in a unit of chunk. The run time algorithm to obtain the optimal $\pi[r]$ is summarized in Algorithm 1. At the start of a new *request event*, update Q_e, Q_d, and Q_r, and estimate $\tilde{r}_l$ and $\tilde{r}_w$ (line 5). For all the possible $\pi[r]$ (lines 6–9), check whether π satisfies the second constraint, and if π is feasible, calculate DPP_r (lines 10–11). The π minimizing DPP_r is selected as the optimal request policy (line 13).

Algorithm 1 Optimal chunk request policy determination

```
Initialize:
1: $Q_e \leftarrow 0$, $Q_d \leftarrow 0$, $Q_r \leftarrow T_b/t_p$, and $N_{w,prev} \leftarrow 0$
During video playback:
2: while Video plays do
3:     if New request event starts then
4:         $N_{w,prev} \leftarrow \hat{N}_w$, $DPP_{min} \leftarrow \infty$
5:         Update $Q_e$, $Q_d$, $Q_r$, Estimate $\tilde{r}_l$ and $\tilde{r}_w$
6:         for $N_t \in \{1, 2, \cdots, Q_{max}\}$ do
7:             for $N_l \in \{1, 2, \cdots, N_t\}$ do
8:                 $N_w \leftarrow N_t - N_l$, $T \leftarrow t_p(N_{w,prev} + N_l) - T_a$
9:                 for $b \in \{b_1, b_2, \cdots, b_m\}$ do
10:                    if $\pi = \{N_l, N_w, T, b\}$ is feasible then
11:                        Calculate $e_e$, $e_d$, $L$, $\Delta$, and $DPP_r$
12:                        if $DPP_r \le DPP_{min}$ then
13:                            $DPP_{min} \leftarrow DPP_r$, $\pi^{opt} \leftarrow \pi$
14:         Request video chunks according to $\pi^{opt}$
15:     else
16:         Wait for the next start time of request event
```

7 PERFORMANCE EVALUATION

Comparison scheme: Although many schemes for video streaming have been proposed to improve QoE, there have been few studies to consider DASH-based video streaming by using both LTE and Wi-Fi simultaneously. We select the energy-efficient HTTP adaptive streaming algorithm (EE-HAS) proposed in [10] as a comparison scheme. To our best knowledge, EE-HAS is the only scheme that can be implemented as an application without modifying other protocol stacks. EE-HAS considers energy consumption and LTE data usage constraint to determine the optimal bitrate of video chunks. The major difference between REQUEST and EE-HAS is that EE-HAS divides each video chunk into two segments, which are requested over LTE and Wi-Fi in parallel.

Performance metrics: the following metrics are used in this work.

i) *Average video quality:* We measure the average video bitrate during the entire playback time.
ii) *Rebuffering:* Rebuffering occurs if a video buffer becomes empty or some chunks have not been successfully received within the time limit due to abrupt throughput degradation or Wi-Fi disconnection in the middle of a chunk download.
iii) *Amount of energy and LTE data waste:* If a user stops watching a video at time instant t, let the waste of energy and LTE data at t be $E_w(t)$ and $D_w(t)$, respectively. We analyze the effect of energy and LTE data waste on overall power and LTE data usage rate, respectively, by dividing $E_w(t)$ and $D_w(t)$ by t. We refer to $E_w(t)/t$ and $D_w(t)/t$ as the time-normalized energy waste and LTE data waste, respectively.

Parameters: In both simulation and measurement, we use $\alpha = 0.5$ for (10), $Q_{max} = 10$, and $t_p = 4$ (s). Video starts after downloading two initial chunks, i.e., $T_b = 8\ s$. We use $V = 1$ for REQUEST for both prototype-based evaluation and simulation. For the measurement, we use 596-second Big-Buck-Bunny video clip[8] encoded at 20 bitrates, i.e., {0.045, 0.089, 0.128, 0.177, 0.218, 0.256, 0.323, 0.378, 0.509, 0.578, 0.783, 1.009, 1.207, 1.474, 2.087, 2.410, 2.944, 3.341, 3.614, 3.936} (Mbps). For the simulation, each trial runs for 600 s, and uses the same video encoded at 10 bitrates, i.e., {0.23, 0.33, 0.48, 0.69, 0.99, 1.43, 2.06, 2.96, 5.03, 6.00} (Mbps).[9]

[8]http://www-itec.uni-klu.ac.at/ftp/datasets/DASHDataset2014/BigBuckBunny/4sec/.

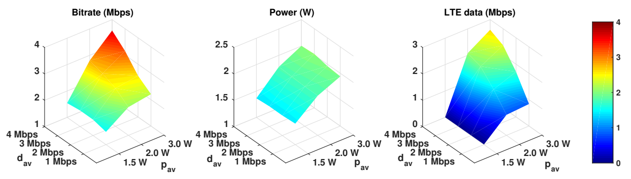


Figure 3: Average bitrate, power, LTE data usage rate of REQUEST. The x-label and y-label of each colormap are power and LTE data constraints, respectively.

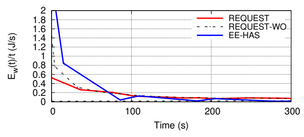
7.1 Prototype-Based Evaluation

We implemented both REQUEST and EE-HAS by modifying the open-source DASH Android application, ExoPlayer. We use Samsung Galaxy S5 smartphone (SM-G900) which runs on Android 5.0. For both schemes, the same power model of SHV-E120 smartphone [16] is used to estimate the energy consumption for chunk download and video playback. [10] We conduct an experiment in a lab environment, where average LTE throughput ranges from 10 to 15 Mbps. We set the maximum throughput of Wi-Fi to 10 Mbps by controlling the QoS option in the setting window of Wi-Fi AP. In the middle of video playback, we degrade Wi-Fi throughput to less than 2 Mbps by adding a contending node with saturated UDP uplink traffic for 180 s.

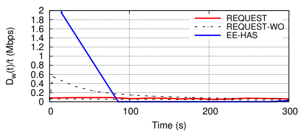
7.1.1 REQUEST with various resource constraints. We first evaluate REQUEST with various power, p_{av} (W), and LTE data, d_{av} (Mbps), constraints, i.e., $\{(p_{av}, d_{av}) | p_{av} \in \{1.5, 2, 3\}, d_{av} \in \{1, 2, 3, 4\}\}$. Fig. 3 shows the average bitrate, power, and LTE data usage rate during the entire playback. During the period when there is no Wi-Fi background traffic from other connected devices, REQUEST fully utilizes Wi-Fi link to save LTE data usage while satisfying the power constraint, and hence, the average LTE data usage is generally less than the LTE data constraint. Obviously, REQUEST can achieve almost the highest bitrate with the highest power and LTE data constraints. Besides, with tight resource constraints, REQUEST tends to select lower average bitrate to satisfy the resource constraints. For all the cases, REQUEST satisfies the constraints.

7.1.2 REQUEST without resource waste consideration. REQUEST considers the expected energy and LTE data waste when calculating the arrival rate of energy and LTE data virtual queues using (3) and (4). To evaluate the resource saving by considering the expected waste, we implemented another version of REQUEST, namely REQUEST-WO, which does not consider the expected waste. Fig. 4 shows the expected waste, i.e., $E_w(t)$ and $D_w(t)$, and normalized waste, i.e., $E_w(t)/t$ and $D_w(t)/t$, when p_{av}=2 W and d_{av}=1 Mbps.

[9]For simulation, we used video encoded with higher bitrate than measurement case since LTE and Wi-Fi throughput measured in real environments was much higher than the highest bitrate used in the measurement.
[10]Our work is independent of power models, i.e., any power model can be used for this REQUEST algorithm.



(a) Time-normalized energy waste.



(b) Time-normalized LTE data waste.

Figure 4: Time-normalized energy and LTE data waste comparison for REQUEST and REQUEST-WO.

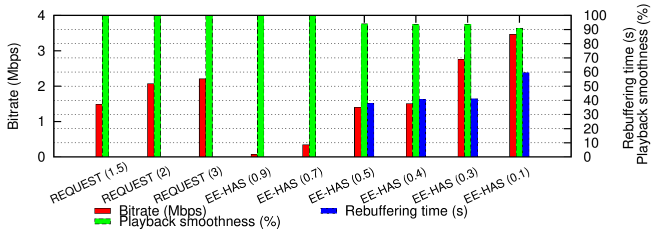


Figure 5: Average bitrate, playback smoothness, and rebuffering time of REQUEST (p_{av}) and EE-HAS (α) with various p_{av} and α. For all the cases, $d_{av} = 1$ *Mbps*.

For EE-HAS, $\alpha = 0.3$. Since EE-HAS is prefetching aggressively, it is more wasteful than REQUEST in the early stage of video playback. Before the video plays for 50 s, both $E_w(t)/t$ and $D_w(t)/t$ of REQUEST are smaller than half of those of REQUEST-WO. REQUEST reduces the waste by selecting lower bitrate and smaller number of chunks to request compared to REQUEST-WO at the beginning of the video. Since the resource waste is relatively high compared to the total resource usage if user stops watching video in a few seconds, REQUEST can save more resources when user quits video early. REQUEST can be a more resource-saving option, and enabling REQUEST-WO can be also an alternative if user simply wants to watch higher video quality with more potential waste.

7.1.3 Comparison study. For comparative evaluation of REQUEST, we run REQUEST and EE-HAS with various p_{av} and α. Fig. 5 presents the average bitrate and total rebuffering time during the entire playback time of the 596-second video. Playback smoothness (%) is defined as $\frac{100 \cdot \text{Total playback time}}{\text{Total playback time + rebuffering time}}$. EE-HAS with small α increases bitrate but suffers from long total rebuffering time, i.e., 41.0 s for $\alpha = 0.3$, and total 59.6 s for $\alpha = 0.1$ when the background traffic starts to degrade Wi-Fi throughput abruptly. Even though EE-HAS with large α, which operates more energy efficiently, does not suffer rebuffering, the average bitrate is quite low, i.e., below 100 kbps with $\alpha = 0.9$. However, REQUEST retains 100% playback smoothness for all the power constraints with over 1 Mbps average bitrate. In addition, the average bitrates of EE-HAS with $\alpha = 0.4$ and REQUEST with $p_{av} = 1.5$ are similar, i.e., over 1.5 Mbps, but EE-HAS shows only 93.6% playback smoothness (40.5 s rebuffering time) while REQUEST provides seamless playback. In summary, in contrast to EE-HAS which provides either too low bitrate without rebuffering or high bitrate with long rebuffering time, REQUEST achieves enhanced quality without rebuffering.

Table 2: Bitrate and rebuffering of EE-HAS ($\alpha = 0.1$) and REQUEST ($p_{av} = 2$) for five traces.

	EE-HAS						REQUEST	
Trace	RT (s)	# ER	ER time (s)	# OR	OR time (s)	Bitrate (Mbps)	RT (s)	Bitrate (Mbps)
1 (office)	4	1	4	0	0	5.27	0	5.17
2 (station)	16.39	3	5.33±1.89	1	0.4	3.03	0.30	5.14
3 (station2)	43.83	3	8±3.27	3	6.61±3.81	2.00	0	2.22
4 (cafe)	12	3	4	0	0	4.06	0	5.17
5 (cafe2)	32.65	3	6.67±1.89	16	0.79±0.16	2.59	0	5.23

7.2 Trace-Driven Simulation

To comparatively evaluate the performance of REQUEST, we implement both REQUEST and EE-HAS using Matlab. For a fair comparison, measured LTE and Wi-Fi TCP throughput traces are used. We first measured the LTE and Wi-Fi TCP throughput using *Iperf* with SM-G900 every second in various places, e.g., office, subway station, and cafeteria. A desktop in a lab is used as a server for *Iperf*.

We use $\alpha = 0.1$ for EE-HAS and $p_{av} = 2$ W for REQUEST. d_{av} is 1.5 Mbps. We classify rebuffering events into two types, i.e., i) rebuffering due to empty video buffer (empty buffer rebuffering) and ii) rebuffering due to absence of video chunks to be played now even though video buffer is filled with video chunks to be played later (out-of-order rebuffering). The out-of-order rebuffering may occur especially with EE-HAS, where a chunk may be divided into two segments that are requested over LTE and Wi-Fi, separately in parallel. If these segments are not received by the player on time, a partially received chunk cannot be decoded and played back, thus causing out-of-order rebuffering. In Table 2, # ER, and ER time denote the number of empty buffer rebuffering events and the average buffering time per event. # OR and OR time are those of the out-of-order rebuffering. RT denotes the total rebuffering time of REQUEST, and bitrate (Mbps) is the average bitrate during playback. For all the cases, rebuffering time of REQUEST is almost zero while EE-HAS suffers from frequent rebuffering events, e.g., 19 rebuffering events and total 32.65 s rebuffering time for the third trace. Bitrate of REQUEST is similar to or higher than that of EE-HAS while REQUEST avoids rebuffering even in mobile and dense environments where LTE and Wi-Fi throughput are often unstable.

8 CONCLUSION

In this paper, we propose REQUEST which utilizes both LTE and Wi-Fi networks to provide seamless video streaming under resource constraints. The proposed chunk request policy of REQUEST ensures that all the video chunks are received even in unstable network environments. REQUEST achieves near-optimal time-average video quality while satisfying time-average resource constraints by adopting Lyapunov optimization framework. Our prototype-based evaluation and trace-driven simulation demonstrate that REQUEST provides seamless video streaming by using both LTE and Wi-Fi links in real-world environments.

ACKNOWLEDGMENT

This work was supported by the National Research Foundation of Korea (NRF) grant funded by Korea government (MSIP) (NRF-2015R1A2A2A01006750) and PraNa project from TEKES, Finland.

REFERENCES

[1] 2014. RFC 7233 (HTTP/1.1 range requests). https://tools.ietf.org/html/rfc7233. (2014).

[2] 2016. Airplug. http://www.airplug.com/solution.php (Accessed: 18 Oct. 2016). (2016).

[3] 2016. ExoPlayer. http://google.github.io/ExoPlayer/. (2016).

[4] 2016. Galaxy S5 Download Booster. http://galaxys5guide.com/samsung-galaxy-s5-features-explained/galaxy-s5-download-booster (Accessed: 18 Oct. 2016). (2016).

[5] 2016. Gpac. https://gpac.wp.mines-telecom.fr. (2016).

[6] Sehyun Bae, Daehyun Ban, Dahyeon Han, Jiyoung Kim, Kyu-haeng Lee, Sangsoon Lim, Woojin Park, and Chong-kwon Kim. 2015. StreetSense: Effect of Bus Wi-Fi APs on Pedestrian Smartphone. In *Proc. ACM IMC*.

[7] Duc Hoang Bui, Kilho Lee, Sangeun Oh, Insik Shin, Hyojeong Shin, Honguk Woo, and Daehyun Ban. 2013. GreenBag: energy-efficient bandwidth aggregation for real-time streaming in heterogeneous mobile wireless networks. In *Proc. IEEE RTSS*.

[8] Jiasi Chen, Amitabha Ghosh, Josphat Magutt, and Mung Chiang. 2012. QAVA: quota aware video adaptation. In *Proc. ACM CoNEXT*.

[9] Savio Dimatteo, Pan Hui, Bo Han, and Victor OK Li. 2011. Cellular traffic offloading through WiFi networks. In *IEEE MASS*.

[10] Yunmin Go, Oh Chan Kwon, and Hwangjun Song. 2015. An energy-efficient HTTP adaptive video streaming with networking cost constraint over heterogeneous wireless networks. *IEEE Trans. on Multimedia* 17, 9 (2015), 1646–1657.

[11] Mohammad Ashraful Hoque, Matti Siekkinen, and Jukka K. Nurminen. 2013. Using crowd-sourced viewing statistics to save energy in wireless video streaming. In *Proc. ACM Mobicom*.

[12] Wenji Hu and Guohong Cao. 2015. Energy-aware video streaming on smartphones. In *Proc. IEEE INFOCOM*.

[13] Junxian Huang, Feng Qian, Alexandre Gerber, Z Morley Mao, Subhabrata Sen, and Oliver Spatscheck. 2012. A close examination of performance and power characteristics of 4G LTE networks. In *Proc. ACM MobiSys*.

[14] Te-Yuan Huang, Ramesh Johari, Nick McKeown, Matthew Trunnell, and Mark Watson. 2014. A buffer-based approach to rate adaptation: Evidence from a large video streaming service. In *Proc. ACM SIGCOMM*.

[15] Kyung-Hwa Kim, Hyunwoo Nam, and Henning Schulzrinne. 2014. WiSlow: A Wi-Fi network performance troubleshooting tool for end users. In *Proc. IEEE INFOCOM*.

[16] Jonghoe Koo, Wonbo Lee, Yongseok Park, and Sunghyun Choi. 2015. PIMM: packet interval-based power modeling of multiple network interface-activated smartphones. In *Proc. ACM e-Energy*.

[17] Kyunghan Lee, Joohyun Lee, Yung Yi, Injong Rhee, and Song Chong. 2013. Mobile data offloading: How much can WiFi deliver? *IEEE/ACM Transactions on Networking (TON)* 21, 2 (2013), 536–550.

[18] Wonbo Lee, Jonghoe Koo, Sungguen Jin, and Sunghyun Choi. 2015. EQ-Video: energy and quota-aware video playback time maximization for smartphones. *IEEE Commun. Lett.* 19, 6 (June 2015), 1045–1048.

[19] Xin Li, Mian Dong, Zhan Ma, and Felix Fernandes. 2012. GreenTube: power optimization for mobile video streaming via dynamic cache management. In *Proc. ACM Multimedia*.

[20] Vivek Mhatre and Konstantina Papagiannaki. 2006. Using smart triggers for improved user performance in 802.11 wireless networks. In *Proc. ACM MobiSys*.

[21] Michael Neely. 2010. *Stochastic network optimization with application to communication and queueing systems*. Morgan & Claypool Publishers.

[22] Michael J Neely. 2013. Dynamic optimization and learning for renewal systems. *IEEE Trans. Automat. Control* 58, 1 (2013), 32–46.

[23] Quoc-Thinh Nguyen-Vuong, Nazim Agoulmine, and Yacine Ghamri-Doudane. 2008. A user-centric and context-aware solution to interface management and access network selection in heterogeneous wireless environments. *Computer Networks* 52, 18 (2008), 3358–3372.

[24] Ishwar Ramani and Stefan Savage. 2005. SyncScan: practical fast handoff for 802.11 infrastructure networks. In *Proc. IEEE INFOCOM*.

[25] Shravan Rayanchu, Ashish Patro, and Suman Banerjee. 2011. Airshark: detecting non-WiFi RF devices using commodity WiFi hardware. In *Proc. ACM IMC*.

[26] Jinghao Shi, Lei Meng, Aaron Striegel, Chunming Qiao, Dimitrios Koutsonikolas, and Geoffrey Challen. 2016. A walk on the client side: Monitoring enterprise WiFi networks using smartphone channel scans. In *Proc. IEEE INFOCOM*.

[27] Iraj Sodagar. 2011. The MPEG-DASH standard for multimedia streaming over the internet. *IEEE MultiMedia* 18, 4 (2011), 62–67.

[28] Qingyang Song and Abbas Jamalipour. 2005. A network selection mechanism for next generation networks. In *Proc. IEEE ICC*.

[29] Kevin Spiteri, Rahul Urgaonkar, and Ramesh K Sitaraman. 2016. BOLA: near-optimal bitrate adaptation for online videos. In *Proc. IEEE INFOCOM*.

[30] Jiyan Wu, Bo Cheng, Chau Yuen, Yanlei Shang, and Junliang Chen. 2014. Distortion-aware concurrent multipath transfer for mobile video streaming in heterogeneous wireless networks. *IEEE Trans. Mobile Comput.* pp, 99 (July 2014), 2607–2620.

[31] Jiyan Wu, Jingqi Yang, Yanlei Shang, Bo Cheng, and Junliang Chen. 2014. SPMLD: Sub-packet based multipath load distribution for real-time multimedia traffic. *J. Commun. Netw.* 16, 5 (Oct. 2014), 548–558.

[32] Jiyan Wu, Chau Yuen, Bo Cheng, Ming Wang, and Junliang Chen. 2016. Streaming high-quality mobile video with multipath TCP in heterogeneous wireless networks. *IEEE Trans. Mobile Comput.* 15, 9 (2016), 2345–2361.

[33] Changqiao Xu, Tianjiao Liu, Jianfeng Guan, Hongke Zhang, and Gabriel-Miro Muntean. 2013. CMT-QA: Quality-aware adaptive concurrent multipath data transfer in heterogeneous wireless networks. *IEEE Trans. Mobile Comput.* 12, 11 (Nov. 2013), 2193–2205.

[34] Zhisheng Yan and Chang Wen Chen. 2016. RnB: rate and brightness adaptation for rate-distortion-energy tradeoff in HTTP adaptive streaming over mobile devices. In *Proc. ACM MobiCom*.

[35] Xiaoqing Zhu, Piyush Agrawal, Jatinder Pal Singh, Tansu Alpcan, and Bernd Girod. 2009. Distributed rate allocation policies for multihomed video streaming over heterogeneous access networks. *IEEE Trans. Multimedia* 11, 4 (June 2009), 752–764.