GreenBag: Energy-efficient Bandwidth Aggregation for Real-time Streaming in Heterogeneous Mobile Wireless Networks

Duc Hoang Bui, Kilho Lee, Sangeun Oh, Insik Shin*
Dept. of Computer Science
KAIST, South Korea
{ducbuihoang,kh.lee,ohsang1213}@kaist.ac.kr, insik.shin@cs.kaist.ac.kr

Hyojeong Shin†
Dept. of ECE
Duke University, USA
h.shin@duke.edu

Honguk Woo, Daehyun Ban
Software R&D Center
Samsung Electronics, South Korea
{honguk.woo,daehyun.ban}@samsung.com

***Abstract*—Modern mobile devices are equipped with multiple network interfaces, including 3G/LTE and WiFi. Bandwidth aggregation over LTE and WiFi links offers an attractive opportunity of supporting bandwidth-intensive services, such as high-quality video streaming, on mobile devices. However, achieving effective bandwidth aggregation in mobile environments raises several challenges related to deployment, link heterogeneity, network fluctuation, and energy consumption. We present GreenBag, an energy-efficient bandwidth aggregation middleware that supports real-time data-streaming services over asymmetric wireless links, requiring no modifications to the existing Internet infrastructure and servers. GreenBag employs several techniques, including medium load balancing, efficient segment management, and energy-aware mode control, to resolve such challenges. We implement a prototype of GreenBag on Android-based mobile devices which hosts, to the best knowledge of the authors, the first LTE-enabled bandwidth aggregation prototype for energy-efficient real-time video streaming. Our experiment results in both emulated and real-world environments show that GreenBag not only achieves good bandwidth aggregation to provide QoS in bandwidth-scarce environments but also efficiently saves energy on mobile devices. Moreover, energy-aware GreenBag can minimize video interruption while consuming 14-25% less energy than the non-energy-aware counterpart in real-world experiments.**

I. Introduction

Real-time multimedia streaming keeps growing in popularity among mobile users, taking up more than 66% of global mobile data traffic by 2017 (up from 51% in 2012)[1]. An ever-growing demand for high-quality videos imposes challenges for mobile multimedia streaming, since it generally requires time-sensitive and bandwidth-intensive delivery. For example, YouTube supports 4K Ultra-HD videos which require around 19 Mbps for streaming smoothly [2], [3]; Vimeo, a popular video sharing site, recommends to use 10-20 Mbps bit rate for uploading full HD 1080p videos [4]. Furthermore, high definition cameras on modern smartphones are capable of recording full HD 1080p videos at bit rate 17-24 Mbps [5], [6], [7]. Since mobile wireless networks, such as 3G, LTE, and WiFi, are typically bandwidth-limited and unreliable by nature, streaming video can be subject to frequent periods of re-buffering, characterized by playback interruptions.

Contemporary mobile devices are equipped with several network interfaces, including 3G/LTE and WiFi. Wireless service providers continue to expand the coverage of 4G LTE networks in several countries all over the world, with unlimited LTE plans available in some countries. With this fast-growing LTE trend, mobile devices are often located within the coverage of LTE and WiFi simultaneously. Unlike 3G, LTE can deliver a bandwidth comparable to or even higher than WiFi in most cases. This offers an attractive opportunity to meet the QoS (Quality of Service) requirement of bandwidth sensitive services, such as video streaming. The simultaneous use of LTE and WiFi links, known as *multi-homing*, enables a new method to increase usable bandwidth for mobile devices, as it can potentially create one logical link via two physical link aggregation (see figure 1).

Bandwidth aggregation in mobile environments raises many challenges. Bandwidth aggregation usually requires cooperation between content providers, proxy servers, and clients, but receiving an agreement among these sides can be a hurdle in reality. The cooperation is needed since aggregating bandwidth of multiple links requires data to be sent across multiple network paths with data striping to reach a mobile device through multiple network interfaces. In order to deploy bandwidth aggregation widely on the Internet, we need to support it on mobile devices without modifications to existing service providers. The second challenge is *out-of-order packet delivery*. LTE and WiFi links are subject to different network characteristics in terms of latency, loss, and bandwidth. This often leads to out-of-order packet delivery when packets are sent throughout different network interfaces. Many applications, including video streaming, require data to arrive in-order, and out-of-order packet delivery can cause excessive delay for real-time applications. This requires effective multi-link packet scheduling to minimize packet reordering. However, the dynamic nature of the Internet paths, particularly, involving unpredictable mobile wireless networks, makes such scheduling complicated. In addition, the simultaneous use of multiple network interfaces can introduce a significant increase in energy use, depending on their own energy consumption characteristics. For example, a considerable amount of energy can be wasted if an LTE interface experiences irregular data transmission such that it stays longer in a high power-consuming waiting state, rather than in an extremely low-power idle state. Since power is critical to battery-operated mobile devices, it entails incorporating interface-specific energy characteristics into multi-link traffic scheduling.

For many years, many approaches have been developed for bandwidth aggregation on multipath communication. Although previous approaches provided good performance in general, very few of them effectively resolved the following three challenges of the mobile environments at the same time: convenient deployment, packet reordering minimization, and energy efficiency. Most of previous approaches focused on the second challenge to provide a robust streaming service that can adapt to dynamic network conditions. For example, a

*A corresponding author.
†At the time of writing this paper, Hyojeong Shin was a Research Fellow at the Department of Computer Science at KAIST.

considerable amount of work was made to extend TCP for multipath support [8], [9], [10], [11]. Another substantial amount of work was performed for multi-path packet scheduling on network proxies [12], [13], [14], [15]. As such, most of the past work inherently required content providers (media servers) to be equipped with new TCP extensions or network proxies to be deployed. Moreover, very few of previous approaches considered the energy efficiency issue subject to real-time constraints.

In this paper, we present GreenBag, a multi-link data-streaming middleware for supporting reliable and energy-efficient real-time data-streaming services via heterogeneous wireless media. For non-intrusive and practical use, GreenBag is devised as a middleware operating on the mobile device, requiring neither a proxy server nor any modification to the existing Internet infrastructure and servers. The mobile-side solution has benefits of leveraging the availability of system information such as the goodput of service, the length of prefetched data stream, packet-receiving condition, and energy availability, in addition to link monitoring. Multimedia applications can request remote files via GreenBag. GreenBag makes two independent wireless connections to a media server via LTE and WiFi, and each connection requests partial segments of the request files based on the network status. GreenBag assembles the file segments and concurrently supplies the in-order data stream to the multimedia applications.

GreenBag devises a client-side asymmetric link management approach to achieve a reliable, well load balanced, and energy-efficient delivery for real-time data streaming, adapting dynamically to varying network conditions. First, GreenBag designs an efficient segmentation method. Since GreenBag concurrently assembles multiple file segments, the length of a segment is critical to performance. A longer segment increases the potential for out-of-order packet delivery. A shorter segment (or a larger number of segments) imposes more networking overheads from multiple TCP connections. The method reduces the overhead through HTTP pipelining and is able to successfully find a reasonably small segment size with little overhead. Second, GreenBag conducts load balancing between two links. Based on the current network measurements through runtime monitoring, it predicts the network conditions in the near future and determines the load ratio between the two links for every segment according to the prediction. Since mobile wireless networks are subject to uncertainty, such predictions are prone to error even for the very near future. GreenBag employs a recovery mechanism; whenever one link finishes its portion within a segment, it checks whether the other link is significantly lagging due to the inaccurate predictions of network conditions in a way that it is expected to miss the QoS requirement. If so, the former link takes over some portion of the problematic link to recover from the lagging. Last, GreenBag devises an early cut-off policy to improve energy efficiency on mobile devices. It opportunistically stops the use of a redundant link when a single-link is able to receive all the remaining data without violating the real-time requirements. GreenBag aims to provide an optimal mode switch between single-link and dual-link modes, in terms of minimizing the energy consumption while meeting the required QoS.

Contributions. The main contribution of this paper can be summarized as follows.

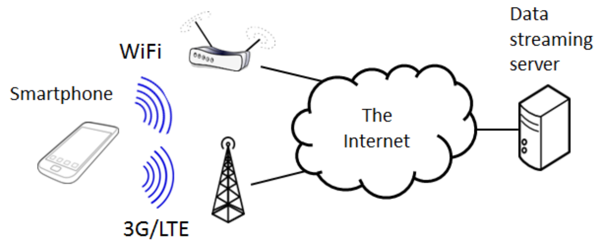


Fig. 1. A scenario of host multihoming on a mobile device with LTE and WiFi networks. The mobile device downloads a video file progressively from a data streaming server with some portion of the file over LTE and the other portion over WiFi.

1) It formulates the bandwidth aggregation problem for real-time video streaming as lexicographic optimization problems that aim to (1) minimize video playback time in order to satisfy QoS requirements, and (2) minimize energy consumption subject to the QoS satisfaction.
2) It provides a design for a multi-link data streaming middleware to support real-time delivery in the most energy-efficient way over unpredictable mobile wireless networks, with the core components of efficient segmentation with HTTP pipelining, medium load balancing with recovery, and energy-aware link-mode control. Since GreenBag is designed to work on top of the widely employed TCP standard protocol, it is compatible with general Internet services without any modification.
3) It presents a prototype of GreenBag implemented on Android-based mobile devices equipped with 3G/LTE and WiFi interfaces. To our best knowledge, this is the first LTE-enabled prototype implementation that demonstrates the effectiveness of bandwidth aggregation for energy-efficient real-time delivery over multiple asymmetric mobile wireless interfaces.

II. Background

A. Challenges

Bandwidth aggregation over multiple wireless links in mobile environments poses many challenges; this section explains some key challenges.

Link heterogeneity and instability. Heterogeneity and instability of wireless links pose a big challenge to the design of effective multi-link packet scheduling policies. Different links often have different performance characteristics in terms of bandwidth, latency, jitter and loss. For example, the bandwidth difference between LTE and WiFi can be largely dependent on the device location, as shown in Figure 2.

Such a large difference in bandwidth can adversely affect the performance of bandwidth aggregation, since it can cause the frequent occurrence of out-of-order packet delivery. Achieving good multi-link packet scheduling, while minimizing packet reordering, essentially requires an accurate prediction of network characteristics of multiple links. However, network instability makes it complicated (if not impossible) to predict accurately the condition of wireless connections even in the very near future. This is because wireless links are inherently unstable and unpredictable as they suffer from

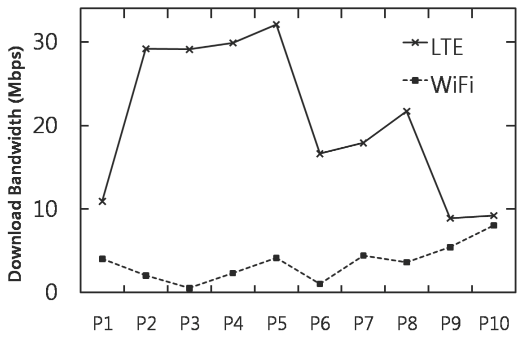


Fig. 2. Significant bandwidth differences between LTE and WiFi across various places in a medium sized city, Daejeon, in South Korea. The bandwidth was measured in 10 random places. P1 to P5 are outdoor places or in coffee shops in the city centre where the LTE signal is strong and WiFi is provided in public; P6 to P10 are located inside a campus, near the edge of the city, where the LTE quality is poor and WiFi connection is from the school's private network.

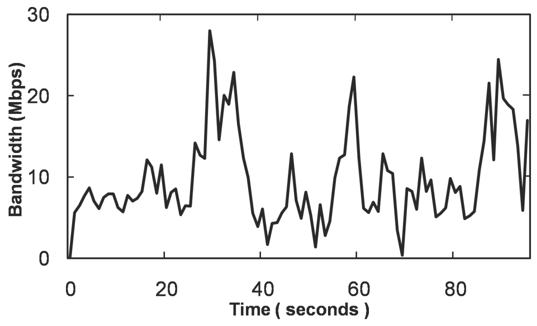


Fig. 3. LTE Bandwidth Fluctuation

random signal interference, congestion, and collisions. This instability becomes even more acute upon devices' mobility. Figure 3 demonstrates rapid fluctuations in LTE bandwidth while the user is walking inside a building in our campus.

LTE/WiFi Power consumption. Power saving is an essential requirement of battery-powered devices. Wireless radio transmission can contribute a significant portion, over 50%, to the total system energy consumption when playing video via HTTP streaming [16]. Moreover, the simultaneous use of LTE and WiFi can impose a significant energy cost. In order to use the energy efficiently, bandwidth aggregation techniques should be designed based on a good understanding of energy consumption characteristics of LTE and WiFi interfaces, as elaborated next.

- LTE power management: LTE network uses *Radio Resource Control* (RRC) protocol for radio resource management. This protocol maintains a single RCC state machine for each mobile device. LTE can be considered to have three RRC states: IDLE, ACTIVE, and TAIL which are shown in Figure 4(a). The device remains IDLE in the long absence of any data traffic. A current state is promoted to ACTIVE when data transmission begins. In the ACTIVE state, a dedicated channel is reserved for the device, and high throughput and low delay is ensured, but at the cost of high power consumption. After data transmission, it is demoted from ACTIVE to TAIL. In the TAIL state, the device shares its channel with other devices and consumes about half of the power in the ACTIVE state. The device remains in the TAIL state until the tail

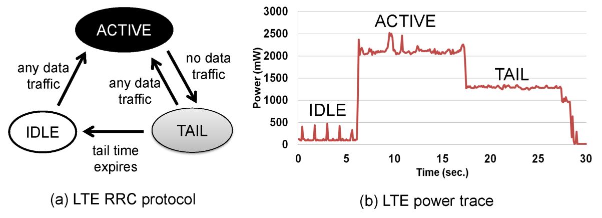


Fig. 4. LTE Power Consumption Characteristics

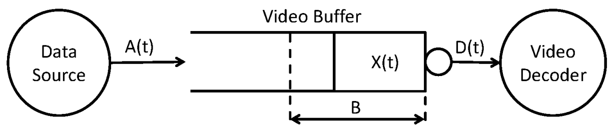


Fig. 5. Video Buffer Model

timer expires, after T_{tail}, and changes to the IDLE state. The IDLE state consumes almost zero power. Figure 4(b) shows an instantaneous power measurement in our experiments on data transfer over LTE.

- WiFi power management: WiFi behaves quite differently than LTE in power management. WiFi usually incurs a high initial cost of associating with an access point (AP). However, because many recent mobile devices use the Power Saving Mode (PSM), the cost of maintaining the association is small. When associated, the energy consumed by data transmission is proportional to the size of the data.

The different energy consumption characteristics should be considered carefully to support an energy efficient multi-link data-streaming service. LTE consumes substantial energy in the long TAIL state after the completion of data transfer, while WiFi is more power efficient than LTE when doing actual data transfer. Therefore, it is not straightforward to maximize energy saving when using multiple links in data transfer since the total energy consumption of a data transfer also depends on the data size and link bandwidth. For example, if the system simply opportunistically offloads data to WiFi interface and disconnects the LTE connection frequently, it can even consume more energy than a non-energy-aware system.

B. Video Player Model

In this paper, we consider a video player which is subject to a QoS requirement such that it downloads a video file of length L (bits) progressively from a media server in order to decode and play the video at bit rate Q (bit/s). The video player typically employs a video buffer in front of its video decoder to avoid many small interruptions to the user. The video buffer can be considered as a queue as shown in Figure 5. For any time t, we denote the amount of data arrived to the video buffer by $A(t)$, the amount of decoded data by $D(t)$, and the amount of data remaining in the buffer by $X(t)$; thus, $X(t) = A(t) - D(t)$.

The video player usually has 2 states, a *playing* state and a *buffering* state. In the playing state, the player keeps downloading the remaining portion of a video file into a buffer and decodes (and plays) the buffered video data concurrently until the buffer becomes empty. On the other hand, in the buffering state, the player only downloads the video file without processing the buffered video data until the buffer is filled with B amount of bits.

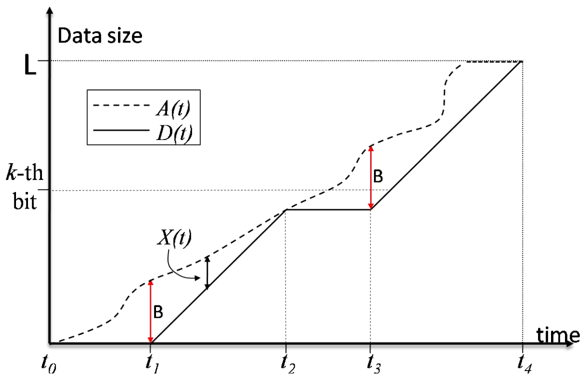


Fig. 6. The video player model with data arrivals, $A(t)$, decoded data, $D(t)$, and buffered data, $X(t)$. At t_1 and t_3, $X(t)$ becomes greater than B, it triggers transitions from the *buffering* state to the *playing* state.

Figure 6 illustrates how a video player works between the two states. In the figure, the player begins at t_0 in the buffering state feeding the buffer. At t_1, the buffer is filled with B amount of bits, and the player switches to the playing state starting to decode and play the video at Q bit rate. At t_2, $X(t_2) = 0$, and the player changes to the buffering state. It resumes the playback at t_3 since $X(t_3) \geq B$. Finally, the video player stops after playing all the length L of the video at t_4, which is the playback time P of the video.

Suppose there is no interruption at all during the playback of a video. This happens when the video player always has non-zero amount of data in the buffer ($X(t) > 0$ for all t) during the playback, and it never stays in the buffering state. In this case, the playback time P is minimized to L/Q and each k-th bit arrives before k/Q. However, the video player may experience one or more interruptions during playback. Let us denote by $I(t)$ a total duration of interruption intervals until t, which is equal to the total duration of the buffering state. For instance, at t_3 in figure 6, $I(t_3) = (t_3 - t_2) + (t_1 - t_0)$. At time t, the amount of decoded data $D(t)$ is equal to $Q \cdot (t - I(t))$. In order to avoid any interruption from t on, all the remaining bits $k \in (D(t), L]$ of a video should arrive into the buffer by a deadline of $k/Q + I(t)$. That is, at time t, the playback time can be minimized when a total delay caused by interruptions is minimized from t on by receiving individual k-th bits prior to their respective deadlines of $k/Q + I(t)$.

C. Problem Statement

Our goal is to support the QoS requirements imposed by real-time video streaming in the most energy-efficient way. As shown in the previous subsection, it is important to minimize the total duration of interruption intervals in satisfying the QoS requirement of real-time video streaming. Thus, our problem can be formulated as optimization problems as follows.

The system divides a video file into N chunks and downloads each chunk through either LTE or WiFi. We denote the chunk allocation vector by $\vec{C} =< c_0, c_1, c_2, \cdots, c_{N-1} >$, where c_i indicates WiFi or LTE, and the chunk size vector by $\vec{S} =< s_0, s_1, s_2, \cdots, s_{N-1} >$ such that $\sum_{i=0}^{N-1} s_i = L$. We also denote the throughputs of LTE and WiFi links over time t by $T_L(t)$ and $T_W(t)$, respectively. Then, $A(t)$ is determined by $T_L(t)$, $T_W(t)$, $\vec{C}$, and $\vec{S}$; $\vec{C}$ and $\vec{S}$ collectively indicate

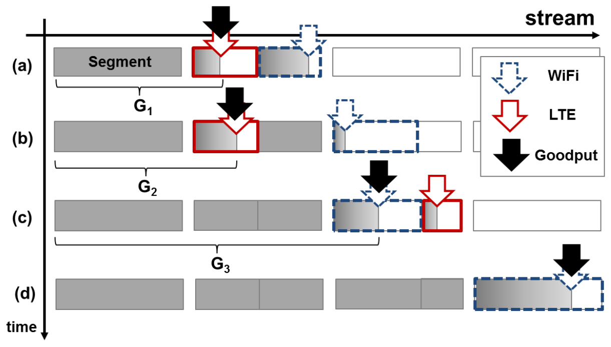


Fig. 7. Multi-link Data Streaming Scheme

which link is allocated to receive a k-th bit, and the arrival time of the bit can be computed over $T_L(t)$ and $T_W(t)$. $D(t)$ is then derived from $A(t)$, since $D(t) = Q \cdot (t - I(t))$ and $I(t)$ depends on $A(t)$ according to the video player model. We define $E(t)$ as the energy consumption of two WiFi and LTE interfaces until time t. Since the energy consumption of network interfaces is determined by throughput and time, $E(t)$ is also determined by $T_L(t)$, $T_W(t)$, $\vec{C}$, and $\vec{S}$. Note that the total playback time P is determined as $D(P) = L$, and the total energy consumption E is determined as $E = E(P)$.

In this paper, we aim to minimize playback time, P, as well as to minimize energy consumption, E. Considering QoS satisfaction is more important than energy saving, we formulate this multi-objective optimization problem as a lexicographic optimization as follows.

#1 Find $\vec{C}$ and $\vec{S}$ that minimize P subject to $T_L(t)$ and $T_W(t)$.

#2 Find $\vec{C}$ and $\vec{S}$ that minimize E subject to $T_L(t)$, $T_W(t)$, and $P = P^\star$, where $P^\star$ is a minimum value of P derived in the previous optimization (#1).

III. Multi-link Data Streaming

This section presents an overview of the proposed GreenBag framework. Figure 7 illustrates the multi-link data streaming process. GreenBag downloads a video file progressively from a remote server, and the file is chunked into multiple file segments. In the figure, segments are represented as boxes, and each box can be further divided into two subsegments. A gray portion indicates the amount of data received. In Figure 7(a), the first segment is complete, and LTE and WiFi are receiving their own subsegments for the second segment. When either of LTE or WiFi finishes receiving its own subsegment, GreenBag arranges the next segment. As an example, WiFi finishes before LTE and moves to the next segment in Figure 7(b).

GreenBag estimates the available bandwidth of both links and determines the sizes of subsegments for medium load balancing. Figure 7(b) illustrates a situation where GreenBag increases the portion of WiFi for the third segment based on its well-performing behavior in the previous segments. The goal of this decision is to have two subsegments finishing at the same time, even considering the remaining portion of LTE in the second segment, in order to avoid out-of-order data delivery.

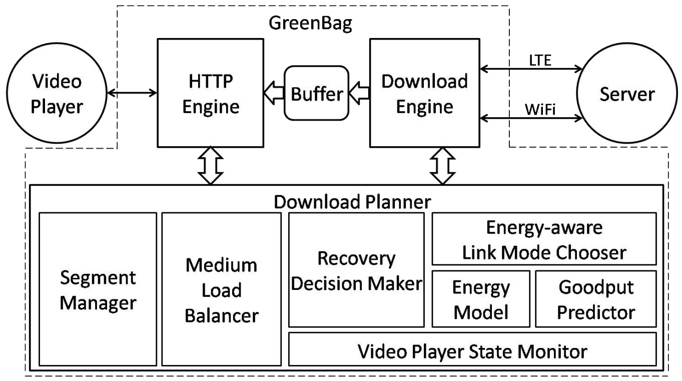


Fig. 8. GreenBag Architecture

When LTE finishes its job for the second segment, it also moves to the third segment, as shown in Figure 7(c). Since the previous segment is complete, the available in-order data expands to the data received by WiFi, represented by G_3. Since GreenBag is capable of using multiple links, it can usually receive data streams much earlier than the requirement of the corresponding applications. This raises a chance to save energy yet meeting the timing requirements imposed by the application. GreenBag keeps track of the available bandwidth and latency of each link. Whenever GreenBag arranges a new segment, it determines whether the remaining data streams can be transferred through a single link in a more energy-efficient way without violating any QoS requirement. If so, the energy-aware mode switch turns off one interface. Figure 7(d) shows a case, where GreenBag uses WiFi only and LTE interface becomes idle.

IV. Design and Implementation of GreenBag

This section describes the design and implementation of GreenBag as a middleware that provides a multi-link data streaming service, aiming at minimizing playback time in the most energy-efficient way in the presence of network fluctuations.

A. System Architecture

GreenBag architecture consists of three main components: an HTTP engine, a download engine, and a download planner, as shown in Figure 8. GreenBag is located between a local video player and a remote server. GreenBag communicates with the local video player through a local connection, over the loopback interface, and with the remote server through two wireless links. In a typical data flow, the video player retrieves a video file to play by sending a standard HTTP request that contains the video file URL to GreenBag. The HTTP engine extracts the URL from the request and passes it to the download planner. The download planner determines how portions of the video file are transferred over the two connections. The download engine then requests each portion of the file over the decided connection, using the HTTP byte-range option. It uses keep-alive connections so that is can send multiple requests for different parts over the same TCP connection. Finally, the HTTP engine sends the downloaded portion from GreenBag's internal buffer to the video player as an in-order byte stream.

B. Download Planner

The download planner addresses various issues in downloading the chunks of a video file according to the multi-link data streaming process described in Section III. Whenever a link is going to finish its subsegment, GreenBag makes the following decisions for further downloading: (1) it first decides if it should recover or not, (2) if not, it then decides the size of the next segment, (3) it then chooses the most energy-efficient link mode, and (4) it finally computes the load balancing ratio between two links.

Segment Manager. In order to resolve the overhead of multiple requests for segments of a file, GreenBag employs HTTP pipelining such that it hides the delay between consecutive requests. It sends a HTTP request for the next subsegment a little bit before it finishes the current subsegment.

With the small request overhead using HTTP pipelining, the segment manager uses a fixed segment size, rather than a variable one. However, the segment manager can determine the segment size with another value in special cases, such as in the end of the file.

The size of a segment, the basic block of GreenBag multi-link data streaming, is critical to performance. A longer segment naturally consists of longer subsegments so it increases the potential for out-of-order data delivery, which typically results in a slower increase in the amount of in-order data, and may lead to violating QoS requirements. On the other hand, a smaller segment size yields a larger number of segments, and generates a larger number of segment requests causing a longer cumulative delay between segments. Section V shows a good range of segment size, and one can be chosen within the range.

Medium Load Balancer. GreenBag uses a heuristic which progressively determines $\vec{C}$ and $\vec{S}$ to maximize the aggregated bandwidth in order to minimize the playback time P. Since the wireless network bandwidth fluctuates rapidly over time, $T_L(t)$ and $T_W(t)$ are unknown, and unpredictable; thus, it is hard to determine the optimal $\vec{C}$ and $\vec{S}$. GreenBag divides a segment into two subsegments, and each subsegment is associated with c_i and s_i in $\vec{C}$ and $\vec{S}$, respectively. It balances load between the two links by using variable subsegment sizes, in order to maximize bandwidth aggregation of two links adapting dynamically to network fluctuation.

The best load balance for a segment is achieved when the two interfaces finish downloading their subsegments at the same time. This way, the maximum amount of data from the later subsegment is added to in-order data at the earliest possible time - the time at which the earlier one finishes.

We need to derive a formula to compute the subsegment sizes of two links in the next segment given its size Z. Suppose that link a is currently finishing its subsegment while the other link b is still downloading its subsegment. Let Z_u represent the total size of all unfinished portions until the end of the current segment. Let Z_a (Z_b) represent the size of the subsegment of link a (link b) in the next segment and the unfinished portion, $Z + Z_u$. Although GreenBag applies HTTP pipelining, i.e. sends requests for the next segment a little bit before the completion of a subsegment, the remaining portion of subsegment of link a is negligibly small. Therefore, Z_a

represents the subsegment size of link a in the next segment. We also denote estimated average goodput of links in the next segment by G_a and G_b. Since the downloading time of a segment is short, G_a (G_b) is estimated to be the average goodput of link a (b) for transferring its current subsegment.

Since GreenBag seeks to finish the two subsegments of the next segment simultaneously, we have the following system of equations:

$$Z_a + Z_b = Z + Z_u \tag{1}$$

$$\frac{Z_a}{G_a} = \frac{Z_b}{G_b} \tag{2}$$

Equation (1) is by definition; equation (2) means the time for downloading each subsegment allocated to each link should be equal. Solving the above equations gives us the subsegment sizes of links a and b in the next segment:

$$Z_a = \frac{G_a}{G_a + G_b} \cdot (Z + Z_u) \tag{3}$$

$$Z_b = Z + Z_u - Z_a \tag{4}$$

After having subsegment sizes, GreenBag allocates the subsegments to the links such that the subsegment with earlier offset is assigned to the faster link.

Recovery Decision Maker. GreenBag can make poor decisions on load balancing due to high network fluctuation and inaccurate bandwidth estimation. For example, the bandwidth of WiFi can drop suddenly due to out of WiFi coverage, and in-order goodput can be restricted by a slower WiFi link while a LTE link is much faster. A subsegment is called a *bottleneck* subsegment if the progress of in-order goodput is directly depending on this subsegment. GreenBag arranges a new segment when a link is about to finish its own subsegment. In the dual-link mode, GreenBag checks if the bottleneck subsegment is significantly lagging in a way that there is a high chance to miss a deadline and cause an interruption with the bottleneck subsegment. If so, GreenBag arranges a recovery such that two links work together to receiving the remaining portion of the bottleneck subsegment.

The principles for recovery are 1) reducing the number of interruptions for QoS satisfaction and 2) reducing the number of recoveries for avoiding overheads. In order to make a recovery decision at t, GreenBag checks the time ($T_P(t)$) to continue playback without interruption based on the amount of buffered data $X(t)$ and the time ($T_R(t)$) required to finish the bottleneck subsegment based on the current bandwidth estimation. GreenBag decides to recover if $w_r \cdot T_P(t) \leq T_R(t)$, where w_r is a weight value.

The recovery mechanism imposes some overhead. Since the HTTP protocol does not allow a client to request for stopping data transfer in the middle of a server's response, GreenBag disconnects completely the connection involving the bottleneck subsegment and reconnects with a new request. Our experiments in section V-A4 show that such an overhead is acceptable.

Energy-aware Link-mode Chooser. GreenBag is aware of LTE and WiFi's asymmetry in terms of energy characteristics so that it can reduce energy consumption. Although LTE can deliver a bandwidth comparable to WiFi, LTE has a long TAIL state which is as long as 11 seconds. Thus, it is energy-beneficial to use LTE to download all the remaining portion of a file continuously to the end, and then stop using the LTE interface.

GreenBag always attempts to minimize energy consumption subject to ensuring no QoS violation. Whenever GreenBag arranges a new segment, it chooses the link mode that consumes the least energy out of three link modes: dual-link mode, LTE-only mode, and WiFi-only mode, in a way that the remaining portion of a video file can be transferred in the link mode without incurring any playback interruption further according to the current bandwidth estimation. Figure 7(d) shows the fourth segment is downloaded in WiFi-only mode.

Since GreenBag cannot have a perfect prediction of bandwidth of each link, it will fall back into dual-link mode immediately for maximum bandwidth, whenever it detects any chance of violating QoS. This is the case when the chosen link mode becomes incapable of transferring the remaining data subject to satisfying deadlines. GreenBag switches to the dual-link mode when $w_e \cdot T_P(t) \leq T_R(t)$, where w_e is another weight value.

Goodput Predictor. The link mode chooser uses predicted goodput values instead of instantaneous goodput since the download time for the remaining file is typically long. The goodput of the link for downloading the remaining file is predicted using exponentially weighted moving average (EWMA), a linear history-based predictor, $\hat{G}_{i+1} = \alpha G_i + (1-\alpha)\hat{G}_i$, with $\alpha = 0.3$, as described in [17], [18].

Energy Model. We derive energy consumption models for LTE and WiFi based on our measurement on a Samsung Galaxy S2 HD LTE phone [19]. To measure energy consumption, we use a Monsoon power monitor [20] which provides instantaneous power consumption at 0.2 ms sampling interval. In order to derive the power consumption of wireless interfaces, we measured the power when data was being transferred with a varying bandwidth of a link with the screen off and no background applications running.

The total energy consumption while downloading/uploading a file is calculated as the sum of the transmission energy consumed by transferring the file itself and the tail energy consumed by staying in the TAIL state. We ignore the promotion energy, from IDLE state to ACTIVE state, of LTE and WiFi as it is too small, around hundreds of mJ, compared with hundreds of thousands of mJ of radio energy in a file download. The transmission energy is modeled as a function of the size of the data and the bandwidth of the link. The transmission energy used to download/upload x Mbits with the bandwidth of y Mbps is described as

$$E_{tx}(x, y) = (\alpha \cdot y + \beta) \cdot \frac{x}{y} \tag{5}$$

The first factor $\alpha \cdot y + \beta$ is the power consumption, and the second factor $\frac{x}{y}$ is the download time. Table I shows the values of α and β derived from our measurement. The tail energy is computed by $E_{tail} = P_{tail} \cdot t_{tail}$, where $P_{tail} = 1350.0\ mW$, and t_{tail} is the time, in seconds, during which the device is in the TAIL state. TAIL state timeout T_{tail} is 11.2 seconds.

Our energy modeling is similar to previous models [21], [22]. Although our derived energy model is device-dependent, there is research for estimating the energy model automatically [23], [24]. However, it is out of scope of our paper.

C. Prototype Implementation

GreenBag. GreenBag is implemented as a system background process and written in C for maximum performance. It runs in background and listens for HTTP requests toward its internal HTTP engine. A video player can retrieve a video file through GreenBag by sending an HTTP request to GreenBag that contains the video URL in a predefined format.

GreenBag provides a Java class for formatting the video URL to GreenBag's predefined format so that developers can use it conveniently it in their Android video player applications. We also implemented a sample video player using Android Media Framework that provides an internal video buffer with the low threshold (B) of 4MB and the high threshold of 20MB.

GreenBag monitors the state of a video player by exploiting the *dumpsys* tool in the Android framework. It periodically queries the number of decoded video frames of the video player. From the number of decoded video frames in a sampling interval, it can estimate whether the player is in the playing or the buffering state. Also, given frame rate and average bit rate of the video, GreenBag can estimate the played time and the remaining time for playing back without interruption ($T_P(t)$) in high accuracy. Current technology, such as *libav* library [25], allows retrieving exact frame rate and average bit rate of a video file given its URL.

Modified Android Framework. We modified the ConnectivityService, a system service of Android, to enable LTE and WiFi interfaces at the same time. Normal Android framework automatically turns off LTE when a WiFi connection is successfully established. Additionally, we made a change to the Android framework such that it configures routing tables correctly when more than one network interface is active.

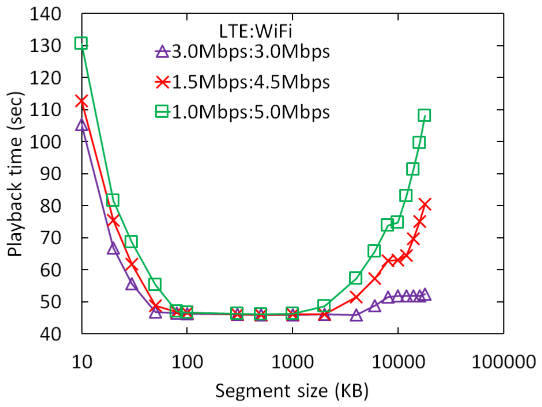
V. Evaluation

We evaluated a GreenBag prototype in emulated and real-world environments. The emulated environment allows us to isolate and analyze the effect of different parameters of GreenBag. The real-world environment tests the efficiency and deployment of the prototype in the real-world environment, in particular, with Android operating system.

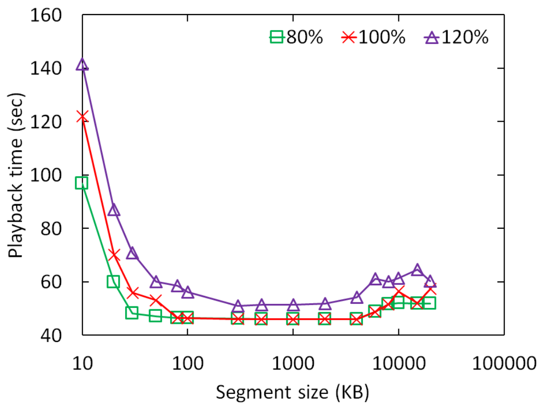
We developed an individual profiler to measure the video interruption time, and energy consumption of GreenBag in both environments. The profiler measures throughput between each network interface of the client and the server, then computes the total energy consumption according to the energy model described in Section IV-B.

	LTE	WiFi
α	16.72	24.19
β	2022.2	360.9

TABLE I
Energy model for data transfers over LTE and WiFi.



(a) Overhead of Segment Size in Different Heterogeneity



(b) Overhead of Segment Size in Different Video Bit Rates

Fig. 9. Overhead of Segment Size

A. Emulated Environment

1) Experiment Setup: The emulated environment consists of an emulated network and an emulated player. The emulated network comprises three computers: a client, a server, and an intermediate node between the client and server. We obtained bandwidth and delay measurements of real-world LTE and WiFi networks, and emulated them by using traffic shaping and network emulation tools, such as Hierarchical Token Bucket and NetEm [26], on the intermediate node. For example, the delay of LTE was set to be higher than WiFi to emulate the asymmetry of the links in reality. In order to create the workload for GreenBag, we implemented the video buffer model as described in Section II-B in the emulated player. Experiment results are averaged over 10 trials. Unless stated, 95-percent confidence intervals, calculated using t-distribution, are omitted from graphs since they are typically very small, around 1%.

2) Segment Size Overhead: In order to understand the effect of segment size on the performance of GreenBag, we did experiments with different segment sizes. The result shows that optimal segment sizes are largely independent of bandwidth heterogeneity and video bit rate. We also note that the minimum video playback time is longer than the video duration since it includes the unavoidable initial start time.

The first set of experiments was performed with different bandwidth heterogeneity. Figure 9(a) shows the playback time of a video with GreenBag in environments, where the total bandwidth of two links is fixed to 6 Mbps with different

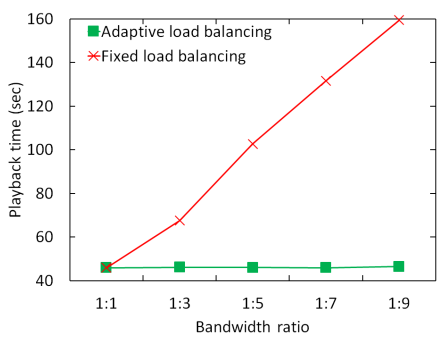


Fig. 10. Effectiveness of Adaptive Load Balancing

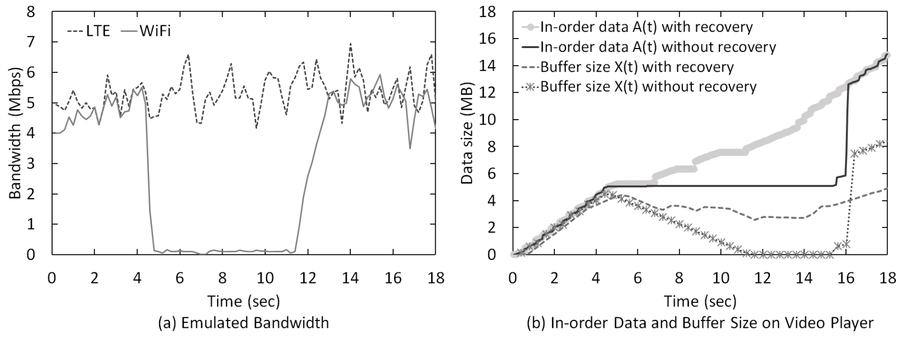


Fig. 11. An Example of the Effectiveness of Recovery Mechanism

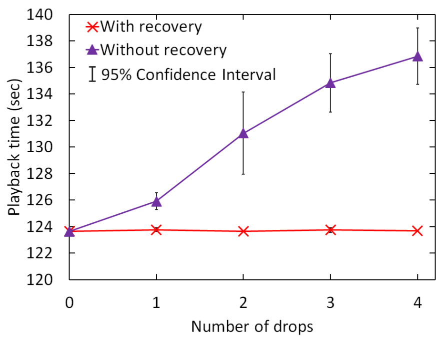


Fig. 12. Effectiveness of Recovery Mechanism

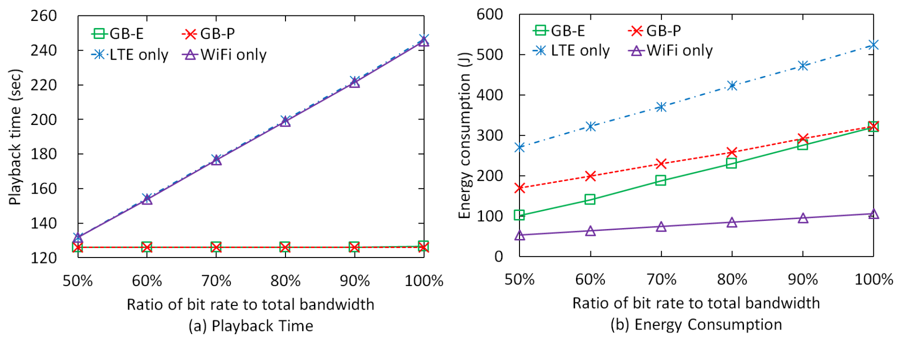


Fig. 13. Effectiveness of Energy-aware Link-mode Switching

bandwidth ratios of 1:1, 1:3, and 1:5. The bit rate of the video is 5.4 Mbps, which is 90% of the total bandwidth, and the video duration is 40 seconds. The result shows the optimal segment size ranges from 100 KB to 1,000 KB regardless of bandwidth ratio. [1] The playback time becomes worse when the segment size gets too small or too large. This is because a very long segment could suffer more from out-of-order data delivery, and a very short segment can impose more overheads in requesting every new segment.

Figure 9(b) shows the effect of segment size over different video bit rate requirements. The experiments are performed over the video bit rates of 80%, 100%, and 120% of the total bandwidth of 6 Mbps; the bandwidth ratio is set to 1:1. The video duration is 40 seconds. Figures 9(a) and 9(b) show a very similar trend that the playback time is smallest when the segment size is from 100 KB to 2,000 KB. In addition, too small or too large segment sizes have significant negative effects on playback time. Based on the experiment results, we use the segment size of 500 KB in the remaining experiments in the emulated environment.

3) Effectiveness of Medium Load Balancing: Figure 10 demonstrates the effectiveness of medium load balancing between two asymmetric links. The performance of GreenBag with the adaptive load balancing scheme is compared with the fixed load balancing case in which the subsegment ratio is fixed to 1:1. Experiments are carried out with a video of 4.8 Mbps bit rate and 40-second duration in the total bandwidth of 6 Mbps with different bandwidth ratios. The figure shows that link heterogeneity in bandwidth has little effect on the adaptive load balancing scheme, while it affects the performance of the fixed load balancing case significantly.

4) Effectiveness of Recovery: Another set of experiments is performed to evaluate the recovery mechanism of GreenBag in the presence of high fluctuation in bandwidth. For example, WiFi bandwidth can decrease or increase rapidly when the user moves around WiFi coverage. Figure 11(a) illustrates a synthesized network bandwidth in which WiFi experiences one high bandwidth fluctuation. Figure 11(b) shows the difference of in-order data and buffer size when using GreenBag with and without the recovery mechanism. With recovery mechanism, the in-order data is smooth so the buffer is not exhausted and there is no interruption. GreenBag recovers on time after it made a wrong decision due to fast bandwidth change. In the other hand, without recovery mechanism, the video is interrupted as the buffer size becomes empty for several seconds. In this case, GreenBag suffers from a long out-of-order data caused by the bandwidth-dropped link so the video buffer cannot tolerate the too long delay of the in-order data.

Figure 12 shows the effectiveness of recovery mechanism over a different number of high bandwidth fluctuations. Each experiment has up to 4 drops, each drop lasts from 6 to 25 seconds. The video used in the experiments requires a bit rate of 110% of the total bandwidth, and is 110 seconds long. Figure 12 shows that the recovery mechanism really helps to recover from inaccurate load balancing decisions due to sudden bandwidth changes. The playback time keeps increasing substantially when GreenBag does not use recovery upon an increasing number of bandwidth fluctuations. On the other hand, the playback time stays stable when GreenBag employs recovery over a different number of fluctuations.

In the experiments, bandwidth drops cause out-of-order

[1] We use kilo binary unit for bytes, i.e. 1 KB = 2^{10} Bytes.

segments which decrease in-order goodput since the in-order received data is limited to the last byte downloaded over the bandwidth-dropped connection. Although the total average bandwidth is still sufficient for streaming the video without interruption, video interruption time will increase if GreenBag does not response appropriately when a bandwidth drop occurs.

5) Effectiveness of Energy Saving: GreenBag aims to minimize playback time to support QoS and then seeks to minimize energy consumption subject to the minimum possible playback time. We ran experiments with two configurations of GreenBag: GB-E and GB-P. GB-E takes care of energy saving as well as supports QoS; GB-P strives to only minimize playback time, without energy saving. The experiment results show that when network condition does not fluctuate too rapidly, GB-E provides the same video playback performance to GB-P while reducing energy consumption.

Figure 13 compares GB-E, GB-P, LTE-only, and WiFi-only in terms of playback time and energy consumption over different video bit rate requirements. The bandwidths of LTE and WiFi are both fixed to 3 Mbps. The video is 120 seconds long and requires different bit rates ranging from 50% to 100% of the total bandwidth. Figure 13(a) shows that GB-E and GB-P have the same playback times over different bit rate requirements, while using a single link only keeps increasing playback times when a higher bit rate is requested. Figure 13(b) shows that GB-E can save 1%-40% more energy than GB-P, even though they provide the same playback times and thereby the same QoS satisfaction. WiFi-only is shown to consume the smallest amount of energy, but it could end up with a very long playback time as shown in Figure 13(a), introducing many playback interruptions and thereby substantial QoS degradation.

B. Real-world Networks

1) Experiment Setup: We ran experiments in real-world LTE and WiFi networks using a Galaxy S2 HD phone with Android 4.0.3. [2] Although we could limit the bandwidth of WiFi using the QoS feature of the WiFi access point, and the bandwidth of LTE by creating cross-traffic from another phone, it is not possible to have full control of the network condition as in the emulated environment. The video used in the experiments requires 6.1 Mbps average bit rate and is 117 seconds long. We used 1000 KB segment size to reduce the number of requests due to high packet loss rate of the real-world networks.

In order to evaluate the characteristics of GreenBag in various cases, we present three representative scenarios: two stationary scenarios, one of which has a higher bandwidth of LTE than that of WiFi *(Stationary #1)* and the other of which is vice versa *(Stationary #2)*, and one mobile scenario, where the bandwidths of WiFi and LTE are subject to fluctuation due to user mobility (*Mobile*). Experiment results are averaged over at least three trials, and the confidence intervals are computed using t-distribution. Table II summarizes the characteristics of the scenarios.

[2] We provide video demonstrations of GreenBag prototype at http://cps.kaist.ac.kr/research/RTSS13_GreenBag/. The demonstrations show effectiveness of multi-link data streaming and energy savings.

	Stationary #1	Stationary #2	Mobile
Average LTE bandwidth (Mbps)	5.16	4.07	4.72
Average WiFi bandwidth (Mpbs)	3.85	5.00	4.94
Total average bandwidth (Mbps)	9.01	9.07	9.66

TABLE II
AVERAGE BANDWIDTH IN EXPERIMENTS IN REAL-WORLD ENVIRONMENT

2) Experiment Result: Figure 14 compares four configurations of GB-E, GB-P, LTE-only, and WiFi-only in the three scenarios. The playback time, shown in Figure 14(a), demonstrates the efficiency of GreenBag in reducing video interruption time by aggregating bandwidth of LTE and WiFi. In all three scenarios, GB-E's playback time is nearly the same as GB-P's, and lower than playback time of both LTE-only and WiFi-only cases. Because the video bit rate is higher than the bandwidth of LTE or WiFi, using LTE or WiFi alone suffers from high interruption time. From energy consumption data shown in figure 14(b), we can see that GB-E consumes less energy than GB-P and LTE-only cases, although higher than WiFi-only case. At the 4-5 Mbps bandwidth in the scenarios, WiFi consumes significant lower energy while LTE consumes much more energy than other cases.

In the Stationary #1 scenario, WiFi's playback time in this scenario is 44-70 seconds longer than that of other cases. Although WiFi can save a lot of energy it could introduce many playback interruptions with long delays. This would be unacceptable from the user experience perspective, since it degrades QoS significantly.

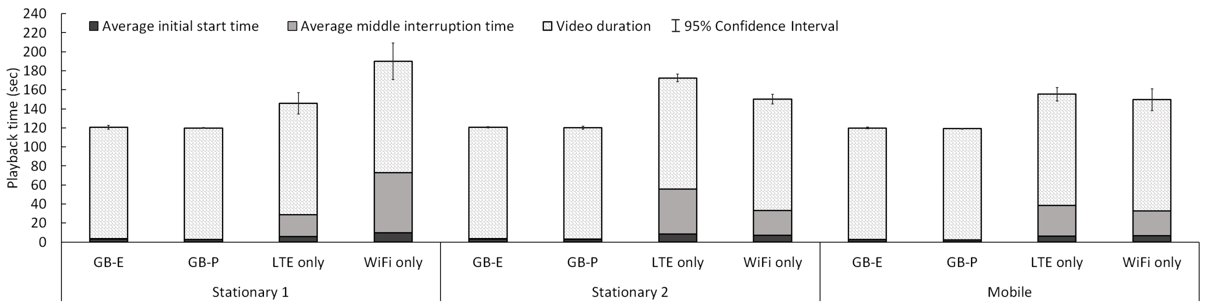
In the Stationary #2 scenario, GB-E saves more energy compared with GB-P than in Stationary #1. Energy saving of GB-E compared with GB-P is 24% in Stationary #2 while it is 14% in Stationary #1. This can be explained by that Stationary #2 scenario has similar total bandwidth with but more WiFi bandwidth than Stationary #1 scenario.

In the Mobile scenario, the mobile phone moved around causing WiFi bandwidth fluctuations. WiFi bandwidth dropped to a very low level, around hundreds of kbps, for 5-10 seconds when the mobile phone moved to a weak WiFi signal region. In spite of suffering from the bandwidth drop of WiFi link, GreenBag recovered correctly so there was no middle interruption time. Also, GB-E still showed a good energy saving in this scenario while there is no difference of playback time between GB-E and GB-P. Because of higher total bandwidth of LTE and WiFi, the energy saving of GB-E to GB-P which is 25% is even slightly higher than Stationary #2.

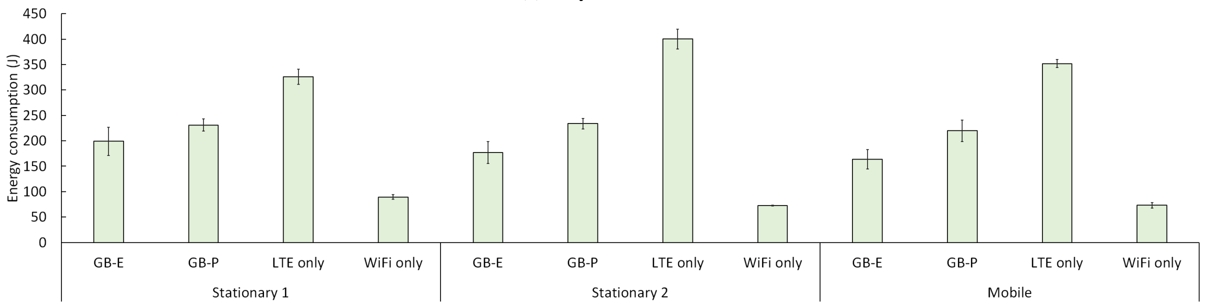
VI. RELATED WORK

Bandwidth aggregation has been an active research topic for many years. Different approaches were proposed at various layers. In particular, the transport layer of the network stack has attracted many techniques for multi-homing. Several efforts [8], [9], [10], [11], [27] have been made to extend TCP/UDP for the capability of using multiple paths simultaneously, and MPTCP (MultiPath TCP) [10], [11] is the most recent and promising TCP variant.

An MPTCP connection (main flow) consists of multiple independent TCP connections (sub-flows). The main flow feeds packets to sub-flows according to the principle of smallest average RTT first, and packets on each sub-flow are striped according to the congestion window of their own sub-flow.



(a) Playback Time



(b) Energy Consumption

Fig. 14. Performance of GreenBag in Real-world Scenarios

While MPTCP has been demonstrated to improve reliability and throughput with multiple paths [10], it has yet to explore how congestion controls should interact between the main flow and sub-flows in order to resolve out-of-order packet delivery for throughput maximization, in particular, over asymmetric lossy links [10].

Similar to MPTCP, MPRTP (Multipath Real-time Transport Protocol) [27] adds multi-path support to RTP, which is typically built on UDP. MPRTP mainly aims to support video streaming with low buffer size requirements. An MPRTP sender distributes packets in an RTP session over multiple subflows, basing on RTCP feedback per subflow from the receiver. However, MPRTP requires support on servers to exploit its capability, and does not take the energy consumption into account.

Many approaches have been introduced for multi-path packet schedulers on network proxies, aiming at minimizing the number of out-of-order packet delivery for effective bandwidth aggregation. A popular scheduling policy is Earliest Delivery Path First (EDPF) [12]. The scheduler estimates the delivery time from the proxy to a client over each path according to the available bandwidth and delay time of each link. The scheduler then assigns a packet to the path with the shortest delivery time. This helps to mitigate out-of-order packet delivery at the client. Many EDPF variants were proposed for different environments, such as lossy links [15], time-slotted networks [14], and generic video encoding formats [13]. PRISM [28] has been introduced as another proxy-based transport layer technique for mobile community networking, where multiple multi-homed mobile devices in close proximity collaborate in utilizing their network links together.

All the above transport layer approaches inherently require changes to the existing Internet infrastructure and/or servers for deployment; from a practical viewpoint, on the other hand, application-level approaches were also proposed for client-side solutions for easier deployment. A recent study [29] is most closely related to ours, sharing the objective of supporting real-time video streaming over multiple links. This recent study presents an application-specific approach that distributes HTTP requests for video streaming across different links in proportion to their respective available bandwidth. However, this study did not consider satisfying QoS requirements and minimizing energy consumption simultaneously. OPERETTA [30] schedules each connection-oriented stream to a network interface, and each packet-oriented stream over different network interfaces in order to increase throughput and/or save energy. However, a connection-oriented stream, such as a TCP connection, cannot be re-assigned to other interfaces once it is assigned to an interface. Moreover, OPERETTA requires support on the server side for distributing packets of packet-oriented streams over multiple network interfaces. It also does not address real-time issues of real-time streaming applications, load balancing in rapidly fluctuating mobile wireless networks, and long TAIL state energy of cellular networks.

MultiNets [31] aims at seamless switching between 3G and WiFi, which avoids interruptions to applications, in order to save energy, offload data, and/or improve bandwidth. It saves energy by switching to 3G when there is no data transfer since WiFi AP scanning makes WiFi consume more energy than 3G in idle state. However, MultiNets does not take the TAIL state overhead of 3G networks into account. Although MultiNets improves throughput by switching to the higher-bandwidth network, estimated by the network signal strength, it does not consider bandwidth aggregation, so it cannot ensure QoS in the case the bandwidth of 3G and WiFi alone is insufficient.

GreenBag can be differentiated from all the above approaches in the following ways. (1) GreenBag does not require any support on existing servers and changes to the existing Internet infrastructure. (2) GreenBag is designed to conserve energy in aggregating bandwidth subject to QoS constraints, while most existing approaches paid little attention to power consumption especially the TAIL state overhead of 3G/LTE networks. (3) GreenBag comes with an LTE-enabled prototype on mobile devices that demonstrates its effectiveness for real-time video streaming.

VII. Conclusion

In this paper, we present GreenBag, a multi-link data streaming middleware that operates in the mobile device without requiring any changes to the existing infrastructure and servers. GreenBag employs various techniques, including efficient segmentation, medium loading balancing with recovery, and energy-efficient mode control, in order to satisfy QoS requirements in most energy-efficient manner, adapting dynamically to network fluctuations. We implemented a prototype of GreenBag on Android-based smartphones, which provides energy-efficient bandwidth aggregation service for real-time video streaming over LTE and WiFi interfaces. Our real-world experiment results show that GreenBag is advantageous in satisfying QoS requirements even in situations where neither LTE nor WiFi meets the requirements. The results also show that GreenBag is able to conserve energy effectively, consuming 14%-25% less energy compared to the (non-energy-aware) throughput maximization case.

In this paper, GreenBag is designed for supporting bandwidth aggregation for non-interactive real-time streaming. We plan to extend GreenBag for interactive real-time streaming applications, such as Skype and FaceTime.

Acknowledgement

This work was supported in part by BRL (2009- 0086964), BSRP (2012-R1A1A1014930), NCRC (2012-0000980), KEIT (2011-10041313), and KIAT (M002300089) funded by the Korea Government (MEST/MSIP/MOTIE), and Samsung Software R&D Center.

References

[1] C. Systems, "Cisco visual networking index: Global mobile data traffic forecast update, 2012-2017."

[2] "What's bigger than 1080p? 4K video comes to YouTube," [Online; accessed 06-Sep-2013]. [Online]. Available: http://youtube-global.blogspot.kr/2010/07/whats-bigger-than-1080p-4k-video-comes.html

[3] "4k resolution," [Online; accessed 06-Sep-2013]. [Online]. Available: http://www.youtube.com/playlist?list=PL5BF9E09ECEC8F88F

[4] "Video compression guidelines," [Online; accessed 1-May-2013]. [Online]. Available: http://vimeo.com/help/compression

[5] B. Klug, "Samsung galaxy S 4 review," [Online; accessed 1-May-2013]. [Online]. Available: http://www.anandtech.com/show/6914/samsung-galaxy-s-4-review/7

[6] B. Klug, "The HTC One review," [Online; accessed 1-May-2013]. [Online]. Available: http://www.anandtech.com/show/6747/htc-one-review/8

[7] A. L. Shimpi and B. Klug, "Apple iPhone 4S: Thoroughly reviewed," [Online; accessed 28-August-2013]. [Online]. Available: http://www.anandtech.com/show/4971/apple-iphone-4s-review-att-verizon/14

[8] H.-Y. Hsieh, K.-H. Kim, Y. Zhu, and R. Sivakumar, "A receiver-centric transport protocol for mobile hosts with heterogeneous wireless interfaces," in *ACM MobiCom*, 2003.

[9] M. Zhang, J. Lai, A. Krishnamurthy, L. Peterson, and R. Wang, "A transport layer approach for improving end-to-end performance and robustness using redundant paths," in *USENIX ATC*, 2004.

[10] S. Barre, C. Paasch, and O. Bonaventure, "MultiPath TCP: From theory to practice," in *IFIP Networking, Valencia*, May 2011.

[11] C. Raiciu, C. Paasch, S. Barre, A. Ford, M. Honda, F. Duchene, O. Bonaventure, and M. Handley, "How hard can it be? designing and implementing a deployable multipath TCP," in *USENIX NSDI*, 2012.

[12] K. Chebrolu and R. R. Rao, "Bandwidth aggregation for real-time applications in heterogeneous wireless networks," *IEEE Transactions on Mobile Computing*, vol. 5, no. 4, pp. 388–403, 2006.

[13] D. Jurca and P. Frossard, "Video packet selection and scheduling for multipath streaming," *IEEE Transactions on Multimedia*, vol. 9, no. 3, pp. 629 –641, 2007.

[14] J. Fernandez, T. Taleb, M. Guizani, and N. Kato, "Bandwidth aggregation-aware dynamic qos negotiation for real-time video streaming in next-generation wireless networks," *IEEE Transactions on Multimedia*, vol. 11, no. 6, pp. 1082 –1093, 2009.

[15] M.-F. Tsai, N. Chilamkurti, J. Park, and C.-K. Shieh, "Multi-path transmission control scheme combining bandwidth aggregation and packet scheduling for real-time streaming in multi-path environment," *Communications, IET*, vol. 4, no. 8, pp. 937 –945, 21 2010.

[16] X. Li, M. Dong, Z. Ma, and F. C. Fernandes, "GreenTube: power optimization for mobile videostreaming via dynamic cache management," in *Proceedings of the 20th ACM international conference on Multimedia*. ACM, 2012, pp. 279–288.

[17] Q. He, C. Dovrolis, and M. Ammar, "On the predictability of large transfer TCP throughput," in *SIGCOMM*, 2005.

[18] M. Mirza, J. Sommers, P. Barford, and X. Zhu, "A machine learning approach to TCP throughput prediction," *Networking, IEEE/ACM Transactions on*, vol. 18, no. 4, pp. 1026–1039, 2010.

[19] "Samsung Galaxy S2 HD LTE," [Online; accessed 28-August-2013]. [Online]. Available: http://www.samsung.com/ca/consumer/mobile/mobile-phones/smartphones/SGH-I757ZKMBMC

[20] "Monsoon power monitor," [Online; accessed 19-September-2012]. [Online]. Available: http://www.msoon.com/LabEquipment/PowerMonitor/

[21] N. Balasubramanian, A. Balasubramanian, and A. Venkataramani, "Energy consumption in mobile phones: A measurement study and implications for network applications," in *Internet Measurement Conference*, 2009.

[22] 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, pp. 225–238.

[23] F. Qian, Z. Wang, A. Gerber, Z. Mao, S. Sen, and O. Spatscheck, "Profiling resource usage for mobile applications: a cross-layer approach," in *Proceedings of the 9th international conference on Mobile systems, applications, and services*, ser. MobiSys '11, 2011.

[24] L. Zhang, B. Tiwana, R. Dick, Z. Qian, Z. Mao, Z. Wang, and L. Yang, "Accurate online power estimation and automatic battery behavior based power model generation for smartphones," in *Hardware/Software Codesign and System Synthesis (CODES+ISSS), 2010 IEEE/ACM/IFIP International Conference on*, 2010.

[25] "libav, open source audio and video processing tools," [Online; accessed 4-May-2013]. [Online]. Available: http://libav.org

[26] S. Hemminger, "Network emulation with NetEm," in *Proceedings of the 6th Australia's National Linux Conference (LCA2005)*, 2005.

[27] V. Singh, S. Ahsan, and J. Ott, "MPRTP: multipath considerations for real-time media," in *Proceedings of the 4th ACM Multimedia Systems Conference*, ser. MMSys '13. New York, NY, USA: ACM, 2013, pp. 190–201.

[28] K.-H. Kim and K. G. Shin, "PRISM: Improving the performance of inverse-multiplexed TCP in wireless networks," *IEEE Transactions on Mobile Computing*, vol. 6, no. 12, pp. 1297–1312, 2007.

[29] K. Evensen, D. Kaspar, C. Griwodz, P. Halvorsen, A. F. Hansen, and P. Engelstad, "Improving the performance of quality-adaptive video streaming over multiple heterogeneous access networks," in *ACM MMSys*, 2011, pp. 57–68.

[30] K. Habak, K. Harras, and M. Youssef, "OPERETTA: An optimal energy efficient bandwidth aggregation system," in *Sensor, Mesh and Ad Hoc Communications and Networks (SECON), 2012 9th Annual IEEE Communications Society Conference on*, 2012, pp. 121–129.

[31] S. Nirjon, A. Nicoara, C.-H. Hsu, J. Singh, and J. Stankovic, "MultiNets: Policy oriented real-time switching of wireless interfaces on mobile devices," in *Real-Time and Embedded Technology and Applications Symposium (RTAS), 2012 IEEE 18th*, april 2012, pp. 251 –260.