Hindawi Publishing Corporation
Mobile Information Systems
Volume 2016, Article ID 4871629, 18 pages
http://dx.doi.org/10.1155/2016/4871629

Research Article

Optimal Multi-Interface Selection for Mobile Video Streaming in Efficient Battery Consumption and Data Usage

Seonghoon Moon, Juwan Yoo, and Songkuk Kim

Yonsei Institute of Convergence Technology, School of Integrated Technology, Yonsei University, Incheon, Republic of Korea

Correspondence should be addressed to Songkuk Kim; songkuk@yonsei.ac.kr

Received 15 July 2016; Revised 25 October 2016; Accepted 25 October 2016

Academic Editor: Abolfazl Mehbodniya

With the proliferation of high-performance, large-screen mobile devices, users' expectations of having access to high-resolution video content in smooth network environments are steadily growing. To guarantee such stable streaming, a high cellular network bandwidth is required; yet network providers often charge high prices for even limited data plans. Moreover, the costs of smoothly streaming high-resolution videos are not merely monetary; the device's battery life must also be accounted for. To resolve these problems, we design an optimal multi-interface selection system for streaming video over HTTP/TCP. An optimization problem including battery life and LTE data constraints is derived and then solved using binary integer programming. Additionally, the system is designed with an adoption of split-layer scalable video coding, which provides direct adaptations of video quality and prevents out-of-order packet delivery problems. The proposed system is evaluated using a prototype application in a real, iOS-based device as well as through experiments conducted in heterogeneous mobile scenarios. Results show that the system not only guarantees the highest-possible video quality, but also prevents reckless consumption of LTE data and battery life.

1. Introduction

The consumption of long, high-definition multimedia content has increased significantly due to the increasing proliferation of mobile devices with bigger screens and higher-performance hardware [1, 2]. In order to provide a smooth video streaming environment, a steady and reliable wireless network is required. Contemporary mobile devices are equipped with multiple network interfaces—including 3G/LTE and WiFi—in order to adjust for a user's network environment. In reality, however, because heterogeneous wireless networks have different transmission rate fluctuations as well as various other drawbacks including power consumption, link capacity, and monetary costs, mobile users are forced to change their connected network interface manually, in order to adapt to dynamically changing wireless environments.

High-definition multimedia applications also lead to faster consumption of a mobile device's battery life and data quota. Thus, mobile video streaming users must closely observe their usage patterns to avoid degraded playback quality and excessive data costs.

Many countries have expanded their 4G LTE coverage, which allows higher bandwidth than 3G networks. However, wireless service providers often set an expensive price for LTE, and almost no country has adopted an unlimited LTE plan. Even unlimited plans that do exist restrict the amount of highest-speed daily data usage due to the limited capacity of the mobile backhaul. Thus, reckless use of LTE networks for video streaming services creates an enormous burden on users in the form of expensive LTE costs.

Additionally, the power characteristic of 4G LTE networks shows that LTE is much less power-efficient than are other wireless networks [3]. Furthermore, WiFi networks are preferred when consuming multimedia content, since WiFi networks are free (or, at worst, cheap) and provide acceptable download bandwidths. Despite these advantages, WiFi is weak in its limited router coverage and handover problems.

To tackle these issue, we propose a novel method that uses adaptive multi-interface selection over both LTE and WiFi links simultaneously, thus guaranteeing the availability of stable HD video streaming over heterogeneous wireless networks. The proposed system provides not only better quality

of service (QoS)—for example, decreased start-up latency and maximized playback quality—but also the real-time optimization scheme design for an interface selection policy that will be controlled by the system to adjust users' remaining monthly (or daily) *LTE data quotas* and *battery lives*. Specifically, to solve an optimization problem, we formulate binary integer programming (BIP), which takes into account the varying average throughput, power consumption, and data costs in order to allocate a video segment to each network interface in an epoch.

Several efforts have been made to develop bandwidth aggregation over multiple interfaces and multipath networks [4–9]. Although previous approaches have presented acceptable experimental results, there are still a few challenges to solve for real mobile environments. It is well-known that packets arrive out of order if the multiple links have different latencies, losses, or bandwidths. This *out-of-order packet delivery* is more harmful to video streaming services than to other types of services because most approaches used to deliver packets over a multipath network divided video bitstreams into chunks on a temporal basis. Video streaming services require more time-sensitive delivery than other web services. However, if the prior packet for playback was allocated on an unreliable or low-bandwidth path, not only is the packet delayed and quality of playback degraded, but the latency and overhead are also increased due to severe packet reordering.

Moreover, while the average throughput can be enhanced by simultaneously using multiple network interfaces, doing so also increases data and energy usage. Thus, systems must be concerned with a user's cellular data quota and battery life in order to maintain positive user experiences. To resolve these problems, we have designed a multi-interface selection scheme with the adoption of split-layer scalable video coding (SVC) and BIP.

SVC is the scalable extension of H.264/AVC [10] and supports spatial, temporal, and quality scalability. SVC provides scalability at a bitstream level; a video encoded with SVC has several substreams within a layered structure—namely, a base layer (BL) and enhancement layers (ELs). The enhancement layers refer to the base layer and allow the video to be immediately changed to a higher quality.

In our previous study [11], we proved the effectiveness of split-layer SVC video streaming in cloud environments for video-on-demand (VoD) service providers. Split-layer SVC streaming can effectively scale to manage the required channels on each layer (e.g., a base layer or enhancement layers) of various client connections. Moreover, the split-layer SVC model brings a remarkable opportunity to stream video content with multiple interface (e.g., WiFi and LTE) selection according to a separate control that is based on network status. By separately controlling the download link, the system ensures that a download request is assigned to an appropriate interface based on the priority of the video segment.

In addition, due to users' frequent mobility and the uncertain link capacities of the wireless networks, stalling often occurs during video playback. To avoid this, video service users need to manually reduce the video's resolution to that allowed by their download bandwidth. By adopting split-layer SVC encoding, our system dynamically controls the download links according to the fluctuating links' average throughput; it then allows the video quality to be adaptively changed immediately.

Our proposed system transfers packets via multipath networks after splitting SVC encoding into a spatial basis (picture size)—instead of splitting packets in a temporal basis (time-sequential) manner. With this spatial basis splitting, we prevent *out-of-order packet delivery* and provide the highest-possible quality of video streaming within the available aggregated bandwidth. For this study, we implemented a mobile-side solution on an iOS device to make it compatible with split-layer streaming; this solution is a middleware in order to facilitate the downloading base layer and enhancement layers using asymmetrical link management. A detailed description of this system design is provided in Section 4.

In summary, our contributions are as follows. Through a real-time adaptive interface selection, we have developed a video streaming system that has (a) guaranteed maximized video quality that the total aggregate bandwidth can accommodate and (b) cost-effective and energy-aware interface control, which is obtained by optimizing the interface selection policy to adjust the users' LTE data quota and battery usage. Additionally, (c) through client-side request scheduling, the streaming server can avoid bottlenecks or packet scheduling overhead for each connected client. Finally, (d) we have developed a prototype of the adaptive multi-interface selection model on an iOS device via LTE and WiFi networks and evaluated our system in emulated environments with several harsh real-world networks.

The remainder of this paper is organized as follows. In Section 2, we explain our work's background and video streaming model. Section 3 provides our problem statement, and Sections 4 and 5 describe the system design and optimization problem formulation, respectively. We present our experimental setup in Section 6 and the results of our experiments in Section 7. A comparison of our work with related work is presented in Section 8. Finally, we conclude the paper in Section 9.

2. Background

2.1. Split-Layer SVC Streaming. Basically, SVC provides scalability in three dimensions: temporal, spatial, and quality. Temporal scalability can be utilized to diversify the frame rate and to change the number of a group of pictures (GOP). Spatial scalability can be utilized to allow for different picture sizes. Quality scalability allows each picture to have a different signal-to-noise ratio (SNR). The three dimensions result in a temporal ID (TID), dependency ID (DID), and quality ID (QID). The multilayer bitstream is finely divided using various combinations of the three possible IDs. The base layer is the reference for the other layers, which are called "enhancement layers" and which can provide higher-quality video. In a bitstream format, SVC with the standard advanced video coding (AVC) is also composed of a video coding layer (VCL) and a network abstraction layer (NAL). It is possible

to distinguish the base layer and the enhancement layers by analyzing the payload in the NAL unit.

We implemented a layer-splitting module by adding our own C code onto the SVC reference encoding tool—namely, JSVM. A layer-splitting module extracts each layer by parsing the NAL unit header after encoding the video source in SVC. Split layers are sent to a different streaming instance to form a channel. In this way, a video service provider can utilize the advantage of SVC (i.e., scalability without particular transcoding) in accordance with the status of the clients.

In our previous work [11], we evaluated the extent to which split-layer streaming brings advantages to video service providers. The experimental results indicated that the split-layer streaming system brings better resource utilization and network bandwidth conservation compared to conventional SVC streaming.

On the client side, a video bitstream request is managed more efficiently by adopting the advantage of split-layer SVC encoding. Mobile clients manage their downloading requests of specific streaming layers through the streaming nodes' URLs. Furthermore, if mobile clients are in a poor network environment that cannot deliver the highest-quality layer, they may then request a lower-quality layer priory that the total aggregated bandwidth can accommodate. Therefore, clients download the highest-video quality available for acceptable network environments without any playback latency. On the other hand, when conventional SVC streaming is used, client devices cannot recognize which part of the bitstream contains high-quality and which part contains low-quality levels without decoding every single packet; thus, clients cannot decide to download priority packets through specific network interfaces that are in good condition. However, if the packet scheduling process is conducted on the server side, a severe bottleneck occurs, degrading the QoS of all streaming nodes responding to the client's request. For these reasons, split-layer SVC streaming is an optimized streaming system for clients' heterogeneous devices over wireless networks.

2.2. Video Streaming Model. We consider the video streaming model to download packets as *on-off streaming*, shown in Figure 1. At first, for a video V, the video player needs a start-up latency because it must store a predefined *initial buffer*. Thus, we expect that multi-interface streaming will shorten the start-up latency with a higher aggregated bandwidth as compared to single-interface streaming.

In the *on-off streaming*, each part of the video segment is requested in an *ON* period, and the HTTP requesting stops in an *OFF* period. We define a triggering point at which to switch from an *OFF* state to an *ON* state; this triggering point is set to be the amount of data remaining at five seconds for playback. Thus, the video streaming system maintains an *OFF* state until the remaining video bitstream appears in the buffer as $D(t_1) \leq 5$ s without a downloading. Afterward, the system continues to request the next video segment until the buffer is filled to the *buffer* level (e.g., 10 times of the video segment sizes of each layer).

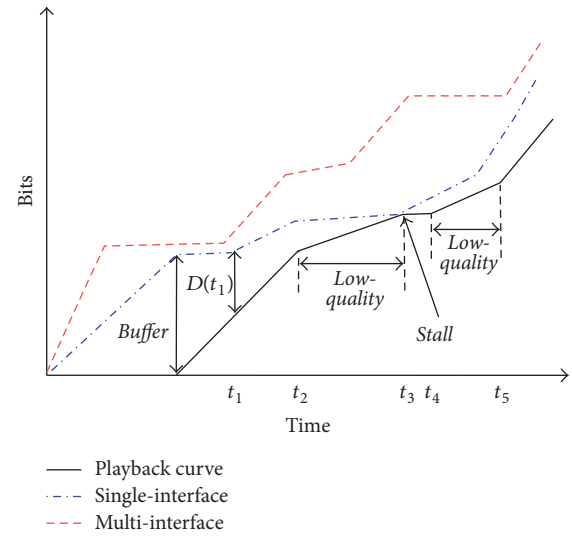


Figure 1: Illustration of a video streaming model. The multi-interface streaming model, which offers a smoother streaming environment than the single-interface streaming model (e.g., LTE-only or WiFi-only) by offering a higher aggregated bandwidth.

A split-layer SVC bitstream is composed of multiple segment sequences, which are displayed at a constant frame rate to the client. However, since the size of each picture varies (in different layers), the required transmission rate varies with time. As mentioned in the above subsection, when the download bandwidth is insufficient to transmit every enhancement layer, the client is unable to play back the highest-possible quality of video content. In Figure 1, if a single connected interface becomes weak, then the average throughput would be lower at time t_2. Thus, the streaming system decides to download the limited-quality layer that the download bandwidth can accommodate, and the client enjoys video content with a lower quality from t_2 to t_3.

At t_3, the amount of buffered data is approximately zero, so the client stalls playback until the interface's download bandwidth is recovered. The streaming system resumes playback at t_4 with a low quality allowed by the interface's throughput. Since the single-interface model is fully recovered enough to download all enhancement layers, the video player decodes the highest quality of video content to the client.

In contrast, the multi-interface streaming model adapts for unpredictable bandwidth fluctuations by measuring, in real-time, each interface's throughput. For example, a WiFi link is preferred by the client with acceptable transmission rates and free WiFi access. The client, however, can move away from router coverage and cause the WiFi connection to be lost or their signal to become weak (as time t_2). At that time, the interface selection system detects a degradation of WiFi throughput and selects 4G LTE to supplement the downloading of all enhancement layers. Therefore, multi-interface streaming provides more stable and satisfactory QoS of high-definition video content than does single-interface streaming. The rest of the paper will describe multi-interface selection in more detail.

3. Problem Statement

As previously mentioned, in order to maintain the quality of video streaming, a mobile client needs to download a higher priority video packet over a better path by adjusting, in real-time, for varying network conditions. Furthermore, that client should be able to play the highest-video quality that the total aggregated bandwidth can accommodate. Also, the proposed streaming system seeks to prevent battery depletion before the entire video has been played back, and it seeks to avoid excessive data consumption by using multiple interfaces. Thus, our problem can be defined as an adaptive interface selection as follows. (The notation used in this paper is summarized in "Important Notations.")

The proposed system divides n split layers into logical segments in each layer with segment size S_n and then downloads this number of segments M_n through either LTE or WiFi links. We denote the total layer by $\vec{L} = \langle 1 == \text{BL}, 2 == \text{EL1}, \ldots, n == \text{EL}(n-1)\rangle$ and the video segments vector by $\vec{V} = \langle V_{1,1}, V_{1,2}, \ldots, V_{1,M_1,\ldots,V_{n,M_n}}\rangle$, where $V_{i,j}$ indicates LTE or WiFi. The bandwidth consumption on each interface can be calculated based on the packet size of the HTTP request, which indicates elements in vector $\vec{V}$. LTE and WiFi bandwidth consumption are denoted by B_{LTE} and B_{WiFi}, respectively.

We denote the expected throughput of each interface for a video segment by $\vec{T}_{\text{Net},i+1}$ (Net $\in$ {LTE, WiFi}); the available bandwidth of $\vec{A}_{\text{Net},i+1}$ is derived from this expected throughput. For example, when the WiFi interface is used for the *fifth* segment, $A_{\text{WiFi},5}$ is initiated to $T_{\text{WiFi},5}$. If one of the layers, with a segment size S_n, uses the available bandwidth, then the available bandwidth is calculated as $A_{\text{WiFi},5} = T_{\text{WiFi},5} - S_n$.

Finally, the system selects the interface to download the next segment by comparing the video bitrate (Q_L) to an available bandwidth $\vec{A}_{\text{Net},i+1}$ with a ratio weight r. The comparing equation is like as follows.

$$\vec{A}_{\text{Net},i+1} \geq r \cdot Q_L. \tag{1}$$

If the calculated link's available bandwidth is larger than the $r \times$ *video bitrate*, the system can select the download link to interface with the Net. To evaluate the performance of this proposed method, we fix the ratio weight at $r = 1.1$. This interface selection scheme will be discussed in more detail in Section 4.2.

As described in the previous section, a higher layer of SVC bitstream can be decoded when the lower layer is delivered well; thus, the system has to *(Problem 1) download from the lowest layer to the higher layers* if the aggregated bandwidth is lower than the sum of the total layers' bitrate.

In addition, we aim to *(Problem 2) minimize LTE bandwidth consumption* B_{LTE} *by maximizing the WiFi bandwidth usage* and to *(Problem 3) avoid the depletion of the user's battery and LTE data quota during playback* by providing an optimal interface selection policy. In the following sections, we describe our solutions to these problems in detail by discussing our system design and formulation of a policy optimizer.

4. System Design

Figure 2 shows the architecture of the proposed system. First, the streaming server splits an SVC-encoded video file based on spatial scalability into multiple split layers. Each split layer is transferred to the virtualized instances in order to build video streaming nodes for the client's HTTP requests. As mentioned in Section 2.1, by using split-layer SVC streaming, mobile clients manage their downloading requests of specific video layers with more explicit priorities (i.e., base layer has higher priority than enhancement layers). For example, if mobile clients are in a low-download bandwidth environment that cannot deliver the highest-quality layer, the split-layer SVC streaming enables user-terminal-driven adaptive streaming to request a higher priority layer's packet through more reliable network interface without decoding process.

On a mobile-client side, the system is composed of three main middleware logics: a video segment manager, an interface selection manager, and a policy optimizer. The policy optimizer includes functions such as a data usage monitor and a battery monitor. The downloaded segments are stored in the system's internal buffer; then, a video player converts the stored split video stream into a single video bitstream, decodes it, and allows the user to watch the video.

4.1. Video Segment Manager. To download the next segment, a video segment manager initiates separate connection to the streaming nodes' URLs; then the video segment manager sends an HTTP request and receives a video segment through a selected interface, using an interface selection manager. After downloading, the video segment manager computes the download link's average throughput in order to determine the interface for the next segment. Here, we present two calculation methods: *INSTANT* and *EWMA*. First, the *INSTANT* method uses the download throughput of the last segment (i.e., BW_i); then, the well-known, linear history-based exponentially weighted moving average (*EWMA*) method predictor is applied to predict the average throughput of the links. The *EWMA* predictor has been widely studied for its use in predicting TCP throughput [12, 13].

Our prediction procedure is presented in Algorithm 1, Lines (1–13). The video segment manager predicts the next average throughput using the *EWMA* algorithm, which is as follows:

$$\widehat{T}_{i+1} = w\,\text{BW}_i + (1-w)\,\widehat{T}_i. \tag{2}$$

In Line (7) of the algorithm, the video segment manager calculates the download bandwidth of the last video segment (BW_i) by dividing the video segment size by the time interval for downloading the segment. Then, from Lines (6–11), the next predicted throughput is computed in the same manner as (2); the result is returned to both the interface selection manager and the policy optimizer. Finally, the video segment manager sends a request for the next segment V_{i+1} in the next epoch through the selected interface by using the interface selection manager. We assume that the server URL is informed in a manifest file sent by the video content provider.

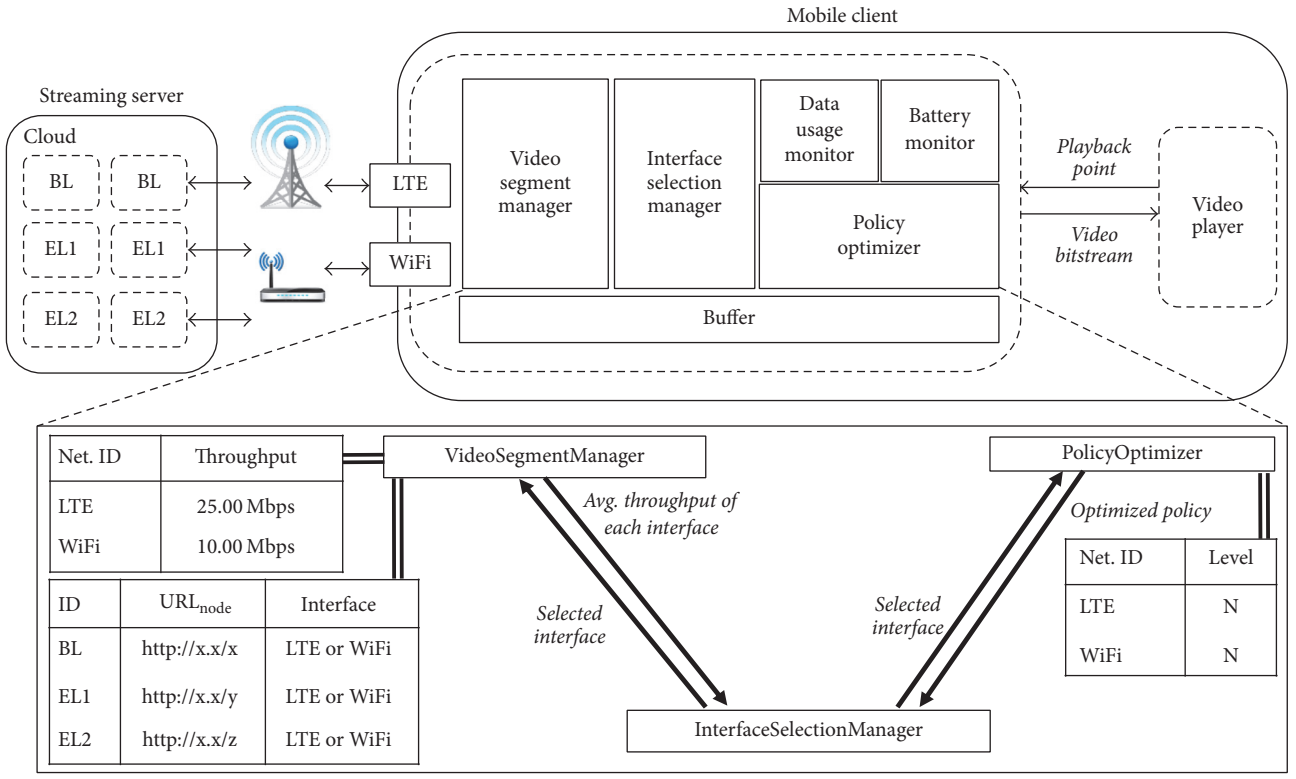


Figure 2: System architecture.

(1) **procedure** VideoSegmentManager(SI_i, URL)
(2) **repeat**
(3) **for all** layer L **do**
(4) **Request** V_i **via** (SI_i, URL)
(5) **end for**
(6) **for all** network interfaces Net **do**
(7) $BW_i \leftarrow \text{sizeof}(V_i)/t_i$
(8) $T_{i+1} \leftarrow w * BW_i + (1 - w) * T_i$
(9) $T_i \leftarrow T_{i+1}$
(10) **return** T_{i+1}
(11) **end for**
(12) **until** disconnect
(13) **end procedure**

(14) **procedure** InterfaceSelectionManager(Q_n, $T_{\text{Net},i+1}$, $\widehat{P}_i$)
(15) Initiate $A_{\text{Net},i+1} \leftarrow T_{\text{Net},i+1}$
(16) Get policy level of P_i(WiFi, LTE) from the Policy-Optimizer
(17) **for all** layer $L \leq P_i$(WiFi, LTE) **do**
(18) **if** $A_{\text{WiFi},i+1} \geq r \cdot Q_n$ **then**
(19) $SI_{i+1} \leftarrow$ WiFi
(20) $A_{\text{WiFi},i+1} \leftarrow A_{\text{WiFi},i+1} - Q_n$
(21) **return** SI_{i+1}
(22) **else**
(23) $SI_{i+1} \leftarrow$ LTE
(24) **return** SI_{i+1}
(25) **end if**
(26) **end for**
(27) **end procedure**

Algorithm 1: Adaptive multi-interface selection.

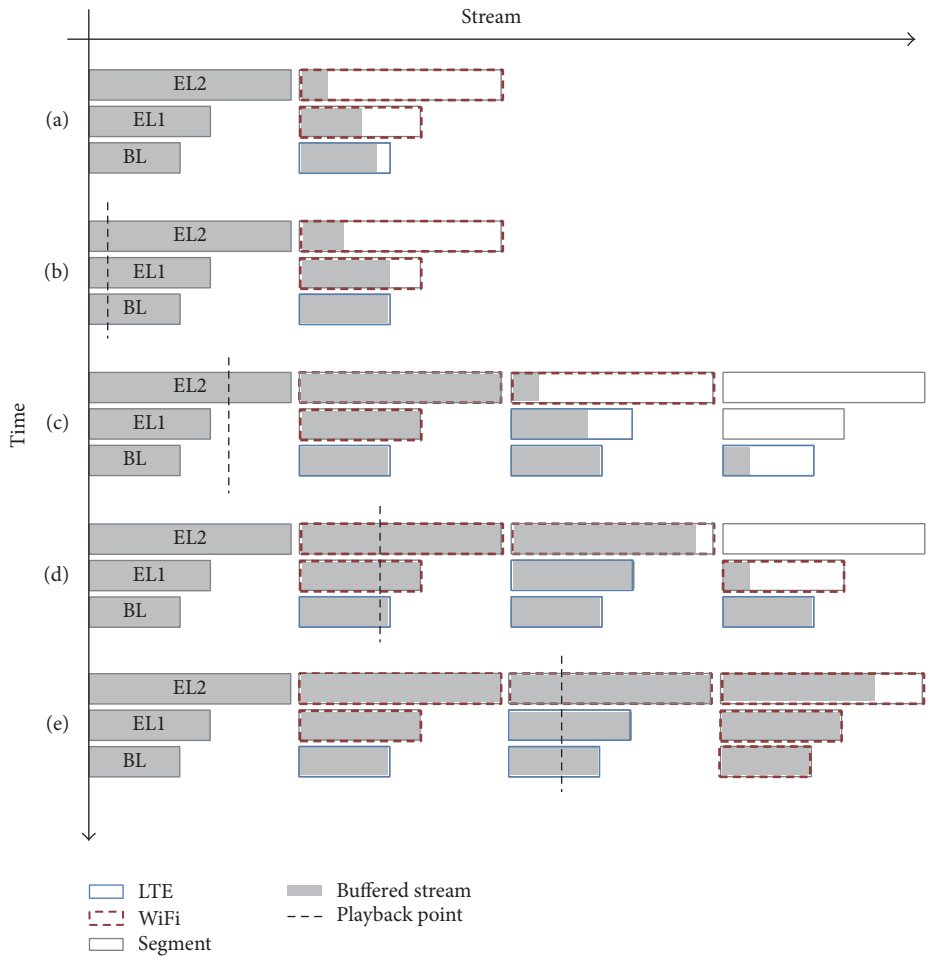


Figure 3: Adaptive interface selection scheme.

For the experiments in this study, we set the smoothness weight at $w = 0.3$.

4.2. Interface Selection Manager. Figure 3 illustrates the multi-interface selection scheme. As shown in the figure, there are three split layers—one base layer (BL) and two enhancement layers (EL1, EL2)—for a scalable video source.

In our previous study [14], we analyzed the impact of the segment size on performance through the experiment. The experimental result showed that the frequent interface changing occurred with the too small segment size setting. On the contrary, if the segment size is too large, the downloading persists until the playback point is approached to buffered level, even the interface's bandwidth has changed. The result also introduced that the optimal segment size was similar to the average video bitrate of each layer; thus, in this paper, we set the segment size of each layer the same as each layer's average bitrate ($S_n = Q_n$). Consequently, in our proposed system, the different buffer sizes are configured on each layer; the buffer sizes are 10 times that of each layer's average bitrate, in other words, the buffer sizes are 10 times the layer's segment size.

From the measured, real-world data taken in South Korea, an LTE link normally has more than 20 Mbps of steady bandwidth [15]. Thus, as shown in Figure 3(a), our system selects LTE as the initial interface for BL, which is a top priority. The other layers (EL1 and EL2) are initially scheduled to be downloaded through a WiFi link. While receiving video packets, an interface selection manager decides simultaneously to keep or to change the download interfaces by comparing the available bandwidth with each layer's bitrate, described by (1); the deciding procedure is presented in Algorithm 1, Lines (14–27).

As a WiFi link's download bandwidth degrades, the playback point approaches the buffered stream, as in Figure 3(b). This means that the available bandwidth is smaller than EL2's bitrate. An interface selection manager monitors the playback point and buffered level every 10 ms; when a gap of less than one second occurs between the playback point and buffered level, the interface selection manager calls back the interface selection procedure to rapidly change the download links.

In SVC streaming, lower layers have priority; hence, the system changes EL1's interface to an LTE link, which can deliver both BL and EL1 while assigning EL2 to a WiFi link, as shown in Figure 3(c). Since EL1 has moved to the LTE link, EL2 can use the entire available WiFi bandwidth.

As shown in the previous subsection, we implement two methods for interface-changing criteria. We also adopt two interface recovery methods. Using the *INSTANT* method, the EL1 link immediately recovers a download interface to WiFi from LTE, after it finishes downloading a segment, as shown in Figure 3(d). Then, as shown in Algorithm 1, the *EWMA* method determines whether to maintain or change the interface by comparing the layer's bitrate with the available throughput. As shown in Figure 3(e), if WiFi is in excellent condition (i.e., $A_{\text{WiFi},i+1} - \sum_{n=2}^{3} Q_n \geq r \cdot Q_1$), the BL segment is also requested via a WiFi link in order to conserve LTE bandwidth consumption.

As has been previously mentioned, we aim to design an optimal interface selection system that adjusts the usage of LTE data and power consumption. Thus, we have designed the policy optimizer to assist the interface selection manager in determining the appropriate interface in each epoch. The interface selection manager, especially, acquires the policy levels of interfaces as an integer vector (e.g., P_i(WiFi = 3, LTE = 2)) from the policy optimizer. In the example given here (Figure 3), the value of the integer indicates that WiFi can be allocated to BL, EL1, and EL2 and that the LTE link can be allocated to BL and EL1. The optimization problem for determining a selection policy will be described in detail in the next section.

5. Policy Optimizer

We now present the policy optimization problem by introducing the objective function used in this study in Section 5.1. We then explain the formulation of the *binary integer programming* (BIP) [16] in Section 5.2.

5.1. Objective Function. Basically, we design a multi-interface system to download the highest-quality video bitstream that is allowed by the links' bandwidths. Moreover, as mentioned in Section 3, two other QoS challenges exist: to avoid excessive LTE data costs and to conserve the device's battery until playback is competed. To address these challenges, we need to find the best interface selection policy in every epoch. To formalize the discretized multi-interface selection problem, we define two binary variables:

$$X_{\text{WiFi}}^{n} = \begin{cases} 1, & \text{the segment manager can download a segment of layer } n \in \vec{L} \text{ via WiFi link,} \\ 0, & \text{otherwise,} \end{cases}$$
$$X_{\text{LTE}}^{n} = \begin{cases} 1, & \text{the segment manager can download a segment of layer } n \in \vec{L} \text{ via LTE link,} \\ 0, & \text{otherwise.} \end{cases} \tag{3}$$

We then propose an objective function, which consists of maximizing the aggregate bandwidth over multiple interfaces, as shown below:

$$\sum_{n \in \vec{L}} \frac{\vec{T}_{\text{WiFi}}}{Q_n} \cdot \frac{1}{n^2} \cdot X_{\text{WiFi}}^{n} + \sum_{n \in \vec{L}} \frac{1}{n^2} \cdot X_{\text{LTE}}^{n}. \tag{4}$$

According to the characteristics of the SVC bitstream, when the video player decodes an upper-layer segment (e.g., EL1), a lower-layer segment (e.g., BL) must be stored in the internal buffer. This means that—as stated in Section 3 *Problem 1*—lower layers are more important than upper layers.

Proposition 1. *To use the layer priority suggested by Problem 1, we multiply a weight value with each binary variable. The weight value is derived by a multiplicative inverse for* n^2.

Proof. To see the sufficiency of (4), let us note a single interface's series. Proposition 1 is proven when the following equation is true:

$$\sum_{n=1}^{k} \frac{1}{n^2} \geq \sum_{n=k+1}^{\infty} \frac{1}{n^2} = \sum_{n=1}^{\infty} \frac{1}{n^2} - \sum_{n=1}^{k} \frac{1}{n^2}. \tag{5}$$

The left side of (5) can be converted as follows:

$$2 \cdot \sum_{n=1}^{k} \frac{1}{n^2} \geq \sum_{n=1}^{\infty} \frac{1}{n^2} = 1 + \sum_{n=2}^{\infty} \frac{1}{n^2}. \tag{6}$$

We then transform the right side of (5) as follows, since the sum is a right Riemann sum that underestimates the area under the integral curve:

$$\sum_{n=2}^{\infty} \frac{1}{n^2} \leq \int_{1}^{\infty} \frac{1}{n^2} = 1. \tag{7}$$

From the above equations, we have

$$\sum_{n=1}^{\infty} \frac{1}{n^2} \leq 2. \tag{8}$$

By rearranging this inequality, we obtain the completed equation (9) as follows:

$$2 \cdot \sum_{n=1}^{k} \frac{1}{n^2} \geq 2 \geq \sum_{n=1}^{\infty} \frac{1}{n^2}. \tag{9}$$

□

Furthermore, *Problem 2* in Section 3 states that we seek to minimize the LTE bandwidth consumption by maximizing WiFi bandwidth usage.

Proposition 2. *To solve Problem 2, we multiply another weighting factor with the terms of X^n_{WiFi}. The weighting factor represents a ratio of the average WiFi throughput to the total size of layers n as follows:*

$$\frac{\vec{T}_{WiFi}}{Q_n}. \tag{10}$$

To select the appropriate network interface in each epoch, we design that the interface selection manager decides the interface through comparison of the predicted throughput and the layers' bitrates, as shown in Algorithm 1, Lines (17–26). Meanwhile, the policy optimizer confirms the availability of a specific interface for each layer by calculating the objective function. For example, if the ratio (10) is larger than one, the expected WiFi throughput could be sufficient for downloading a video segment in layer L, in which case the WiFi link is always preferred over the LTE link.

Overall, the optimal selection policy problem is defined as follows: find the interface selection sets, P_i(WiFi $= \sum_{n\in L} X^n_{\text{WiFi}}$, LTE $= \sum_{n\in L} X^n_{\text{LTE}}$, such that the objective function (4) is maximized with the LTE data usage and the power consumption constraints given in the following subsection.

5.2. Formulation of Binary Integer Programming. The simultaneous use of multiple interfaces risks excessive consumption of LTE data and energy. Thus, we formulate a BIP that jointly decides the following: (1) a set of interface selections that will provide the highest-video quality with maximized aggregated bandwidth, (2) how to best use the user's LTE data without exceeding their limited LTE data quota, and (3) a set of interface selections that will not deplete the user's battery until playback ends. With these definitions, the optimization problem can be formulated as follows:

$$\max \quad \sum_{n\in\vec{L}} \frac{\vec{T}_{\text{WiFi}}}{Q_n} \cdot \frac{1}{n^2} \cdot X^n_{\text{WiFi}} + \sum_{n\in\vec{L}} \frac{1}{n^2} \cdot X^n_{\text{LTE}} \tag{11}$$

$$\text{s.t.} \quad \sum_{n\in L} S_n \cdot X^n_{\text{LTE}} - \vec{T}_{\text{WiFi}} \le \frac{U_{\text{data}}}{R_t} \tag{12}$$

$$\sum_{n\in L} P_w \cdot \frac{S_n}{\vec{T}_{\text{WiFi}}} \cdot X^n_{\text{WiFi}} + \sum_{n\in L} P_l \cdot \frac{S_n}{\vec{T}_{\text{LTE}}} \cdot X^n_{\text{LTE}} \le \frac{U_{\text{battery}}}{R_t}. \tag{13}$$

5.2.1. LTE Data Usage Constraint. Most video service users want to enjoy the highest-possible video quality that the aggregate network bandwidth can accommodate. However, they do not want the expense associated with excessive consumption of LTE data. Thus, we propose the LTE data constraint given in (12), which is based on the users' remaining daily or monthly LTE data quota.

To regulate selection of the LTE link, the optimizer compares the users' available LTE data quota (U_{data}/R_t) with the expected LTE data consumption in every epoch. The expected LTE data consumption is derived using the most recent WiFi average throughput (T_{WiFi}); the remaining playback time is provided by the video player.

5.2.2. Battery Usage Constraint. To address the *Problem 3*, given in Section 3, we have derived a power consumption constraint, given here as (13). A device's battery energy is consumed by packet transmission, decoding, and displaying during video streaming applications; in fact, Li et al. [17] showed that a video packet transmission contributes a significant portion of a device's total system energy consumption during video playback via HTTP streaming. Furthermore, even just doubling the video's resolution, in some cases, increases (albeit slightly) the device's average power consumption [18]. We thus ignore energy variations in the decoding and displaying of video for different layers, because such variation is so small—compared with the amount of radio energy in video segment downloads.

As shown in (13), the battery usage constraint consists of three parts: the power models for data transfer (P_w & P_l), the segment download duration ($S_n/\vec{T}_{\text{Net}}$), and the binary variables. Then, on the right side of (13), we set a term that represents the remaining battery life for the epoch (U_{battery}/R_t).

For this study, we apply the linear-fit model for the power-throughput relationship between LTE and WiFi, which is similar to a well-known previous method [3]. The linear-fit models are as follows:

$$P_w = \alpha_w \cdot \vec{T}_{\text{WiFi}} + \beta_w,$$
$$P_l = \alpha_l \cdot \vec{T}_{\text{LTE}} + \beta_l, \tag{14}$$

where α_w (mW/Mbps), β_w (mW), and α_l, β_l are linear-fit parameters for LTE and WiFi, respectively. In order to determine linear-fit parameters, we use the measurement data of a real device presented by Liu and Wang [19]. Liu and Wang measured the devices' energy profile using "Kill-A-Watt" [20], a power measurement tool that displays the average power of every second. In this study, we build a prototype for iPhone 5; thus, the linear-fit parameters of iPhone 5 are $\alpha_w = 12$ and $\beta_w = 320$ and $\alpha_l = 49$ and $\beta_l = 580$.

6. Prototype Implementation and Experimental Setup

To evaluate our system's performance, we build a prototype application for iPhone 5 written in Swift [21], which is a programing language for iOS applications. The proposed system needs to utilize both LTE and WiFi network interfaces simultaneously; however, the basic iOS SDK does not support multiple interface selection. Thus, we implement a multi-interface system using the open source library CocoaAsyncSocket [22], which uses native iOS system functions to create a user-defined interface connection. It also enables users to send HTTP requests through multiple sockets toward the iOS internal HTTP engine.

We would also like to develop a battery monitor and a data usage monitor. To access the iOS battery status, we implement a monitoring code that is based on the iOS SDK, BatteryStatus

[23]. We assume that the system gets information regarding the user's LTE quota from either the cellular provider or a middleware API supported by the cellular provider. The policy optimizer is also built (specifically to solve the optimization problem of BIP); we design a code using the *branch-and-bound* as a solver, which is a commonly used algorithm for solving NP-hard optimization problems [24].

As mentioned in Section 4.2, we set the initial buffer sizes smaller than default buffer sizes to shorten start-up latency, where the initial buffer sizes are as much as three times of the video segment sizes in each layer; thus, after storing the initial buffer, playback begins. After playback begins, the system configures the default buffer sizes to be 10 times that of each layer's average bitrate. We also define a triggering point at which to switch from an *OFF* (stops to request video segments) state to an *ON* (buffering) state; this triggering point is set to be the amount of data remaining at five seconds for playback. While an SVC-encoded video can be played without enhancement layers, the video quality level is as low as the base layer. Thus, we measure the deadline miss duration, which is closely related to video quality. From the amount of buffered video bitstreams in a sampling interval, we can estimate whether each layer can be played or missed. To measure the possibility of playback in each layer, our system analyzes the state of downloaded video packets every 10 ms.

We implement a streaming server into a Dell server for the experiment. The server is configured with a 16 core 2.67 GHz Intel Xeon E5640, 16 GB RAM, and 4 TB storage. We use a Korean video, "Nobody's Daughter Haewon," for our performance evaluation. The video is encoded in the JSVM software; 1800 frames (with both temporal and spatial scalability) are therein encoded at 30 fps. The video is split into three layers to exploit each of the three spatial levels. Each spatial level is set by the specific sizes that people most often use on their smartphones, tablets, and PCs. Thus, the spatial levels are constructed at BL 320 × 180, EL1 640 × 360, and EL2 1280 × 720. BL, EL1, and EL2 have average bitrates of 1142 Kbps, 1687 Kbps, and 2318 Kbps, respectively. Using our designed code [11], we split a multilayer bitstream by parsing the DID data in the NAL unit header and then stored the split bitstream in separate layers.

First, we analyze and discuss the effects of the policy optimizer regarding various aspects of emulated network environments. Then, we compare the performance of the proposed system with those of the conventional SVC streaming and the single-interface streaming methods in several real-world environments. To experiment in both emulated and real-environment, we use real-world LTE network offered by Korean Telecom (KT). For our emulated WiFi network environment, we create a virtual machine for the intermediate HTTP proxy node. We used the Kernel Virtual Machine (KVM) hypervisor [25] as our virtualized proxy node and set the size of the VM (for experiments) to be the same as the medium type of instance provided by the general IaaS provider. To generate the required timeline-based bandwidth pattern, we design a configuration script for the proxy.

The proxy is written in JavaScript using Node.js, which is efficient for data-intensive, real-time applications [26]. A proxy node calculates the expected number of bytes to pass based on the desired pattern in the configuration script.

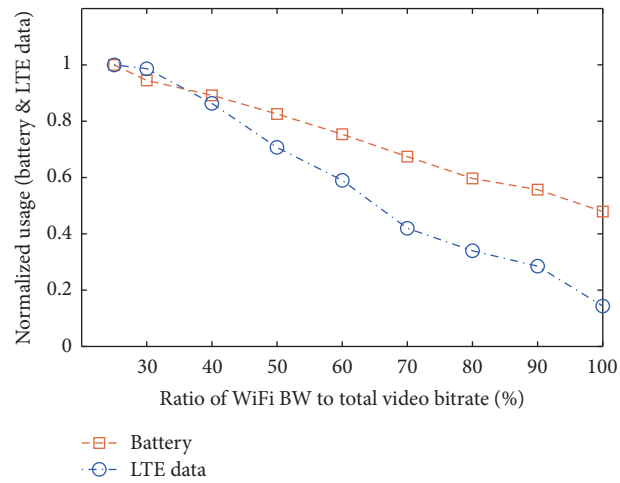


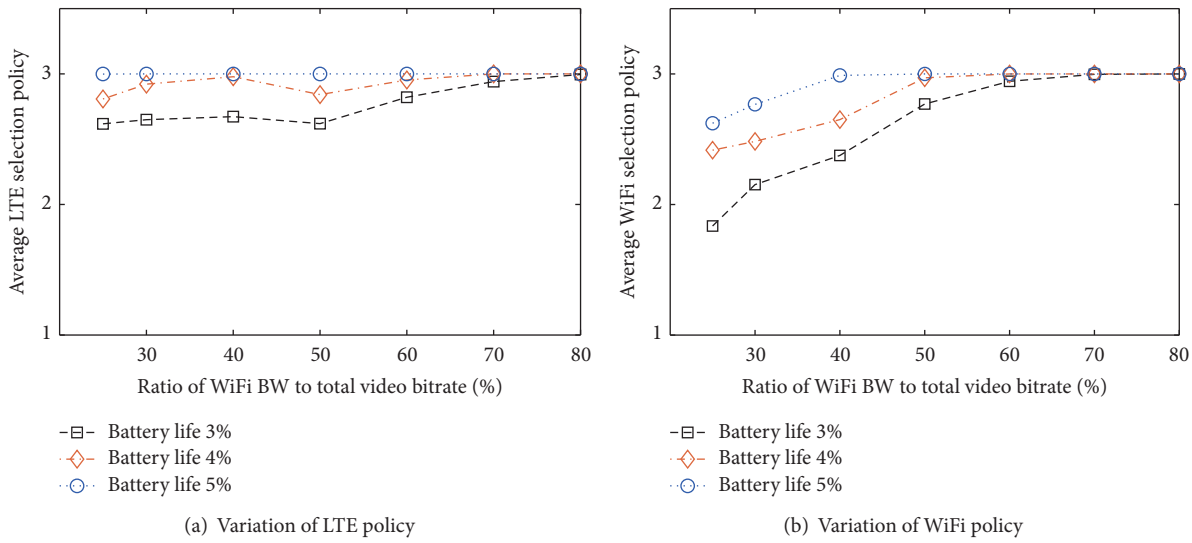
Figure 4: Variation pattern of user's battery and LTE data consumption, according to different ratio of WiFi bandwidth to the total bitrate of all layers.

7. Experiment Result

7.1. Effects of Policy Optimization. In this section, we analyze the effects of the policy optimizer with regard to the users' LTE data quota and battery life. For this evaluation, we designed an environment with multiple wireless networks—for instance, a nonemulated (real-world) LTE bandwidth and an emulated WiFi bandwidth with a growing ratio of 25–80% to the total bitrate of all layers. We derived the LTE data usage by calculating the cumulative size of all segments requested via LTE links. To determine the battery usage required by a single transmission, we computed the power expended during segment transfers across varying link bandwidths; during the measurements, we left the screen off and made sure no background applications were running. We could then measure the battery and data consumption required for 10 min of video streaming.

Figure 4 shows the varying rates of battery and LTE data consumption that occur with simultaneous use of multiple interfaces, where the interface used is based on the ratio of WiFi bandwidth to the total bitrate of all layers. Unsurprisingly, the largest amount of LTE data usage occurs when the WiFi link's bandwidth is low. This result shows that the multi-interface streaming system utilizes the LTE interface actively because of the lacking in WiFi bandwidth. Likewise, battery usage also increases when the streaming user is in low WiFi bandwidth environments. As shown in the power-throughput relationship of each interface, described in Section 5.2, the power consumption increases when the LTE interface is used more intensively for video downloading.

7.1.1. Effects of the Battery Constraint. As we have described, we design a BIP with two constraints—limited battery life and LTE data quota for video transferring—in order to solve



(a) Variation of LTE policy

(b) Variation of WiFi policy

Figure 5: Optimized (a) LTE and (b) WiFi selection policies when the user's device has limited remaining battery life (3, 4, and 5%).

the optimization problem regarding the interface selection policy. Here, we discuss the experimental performance of the proposed method (in terms of LTE data consumption, power consumption, and deadline miss duration) when different amounts of battery life are available within the emulated WiFi network environment. In this case, we configure the users' LTE quota such that enough remains for us to download an entire video bitstream; thus, we are able to evaluate only the effects of the battery constraint on the proposed method.

Figure 5 illustrates the interface selection policies that occurred during 10 min of streaming test with 3, 4, and 5% of the available battery life for the prototype application and various WiFi bandwidths available. The available battery life is set at 3, 4, and 5% because these quantities are appropriate for analyzing the performance of 10 min of video data transferring. In addition, we would not consider the energy variations by displaying and decoding the video bitstream. As mentioned in Section 4.2, if the policy optimizer sets the value (level) of an interface at 1, then the interface selection manager sets that interface only as a base layer; if the interface is set at a value of 2, then the interface selection manager sets that interface as both a base layer and an enhancement layer. By the same logic, the interface selection manager can assign an interface to all three layers by setting the interface selection policy at 3.

As Figure 5(a) shows, when the user's device has limited battery life remaining (e.g., 3 and 4%), that device cannot download entire layers via LTE links for 10 min when the WiFi bandwidth is less than 60% that of the video bitrate. In other words, the user's device (which has only 3 or 4% battery life remaining) will be turned off if the entire layers are downloaded through LTE links in such a case. Meanwhile, the user can use LTE links to download entire layers, regardless of the WiFi throughput, when their device's battery life is 5% or more.

Figure 5(b) presents the average WiFi selection policy for 10 min of video downloading. Because the battery constraint is directly affected by both the interface's power models and the link connection time for downloading (S_n/T_{net}) as shown in (13), when the WiFi bandwidth is less than 40% of the video bitrate, poor levels of selection policy exist across all three battery life scenarios (3, 4, and 5%). Thus, we recognize that a WiFi link's power consumption increases when the downloading session time is extended by a poor WiFi bandwidth.

We also measure the amounts of LTE data consumption and battery usage that occur when a video is downloaded. As shown in Figure 6(a), more LTE data are consumed over low WiFi bandwidths; this aspect of battery usage shows a comparable pattern to that in Figure 6(b). This result shows a similar level of LTE data and power consumption regardless of the user's remaining battery life when the WiFi bandwidth is greater than 60% of the total bitrate; this occurs because the three battery life scenarios result in similar interface selection policies (see Figures 5(a) and 5(b)).

Furthermore, we note that the interface selection policy is also related to the deadline miss duration. Figure 6(c) shows that if all of the interface selection policy levels are low, then a large number of deadline misses occur during playback when the WiFi throughput is less than 40% of the total bitrate. Nonetheless, when our proposed policy optimization scheme was used, not only did a deadline miss occur in layers 1 and 2, but the mobile device also did not shut off until the end of playback.

7.1.2. Effects of the LTE Data Constraint. In order to analyze the effects of the LTE data constraint, we configure the user's remaining LTE quota to be 50, 60, and 70% of the video's total size (386 MB) and perform experiments under the same variable WiFi conditions as those in the previous subsection. To make sure that only effects of the LTE quota are evaluated in these experiments, we set the device's battery life to be large enough to download the entire video bitstream. Figure 7 shows the results of the LTE and WiFi selection policies

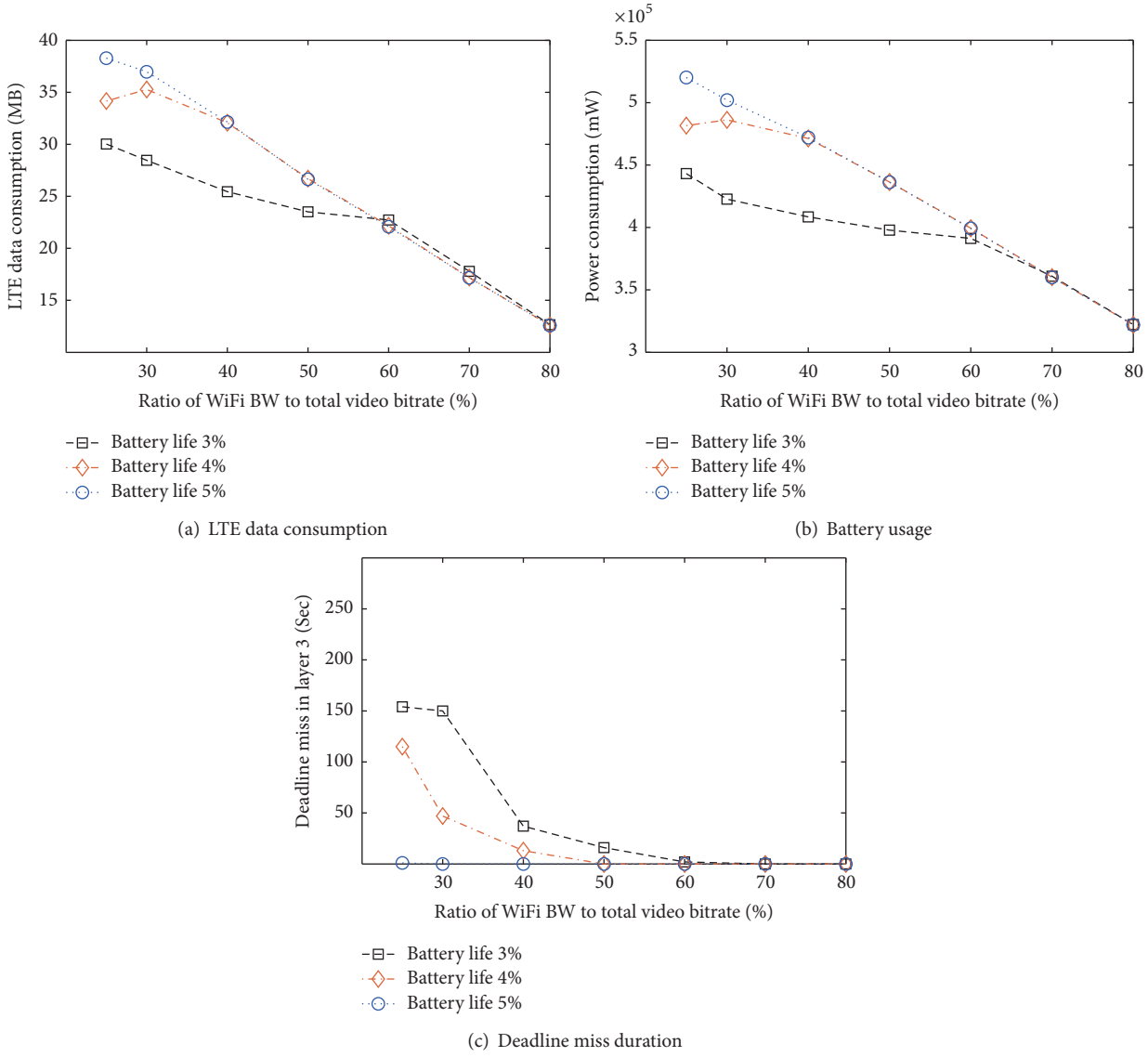


FIGURE 6: (a) LTE data consumption, (b) battery usage, and (c) deadline miss duration according to varying ratios of the WiFi bandwidth to the total bitrate of all layers when limited battery life remains.

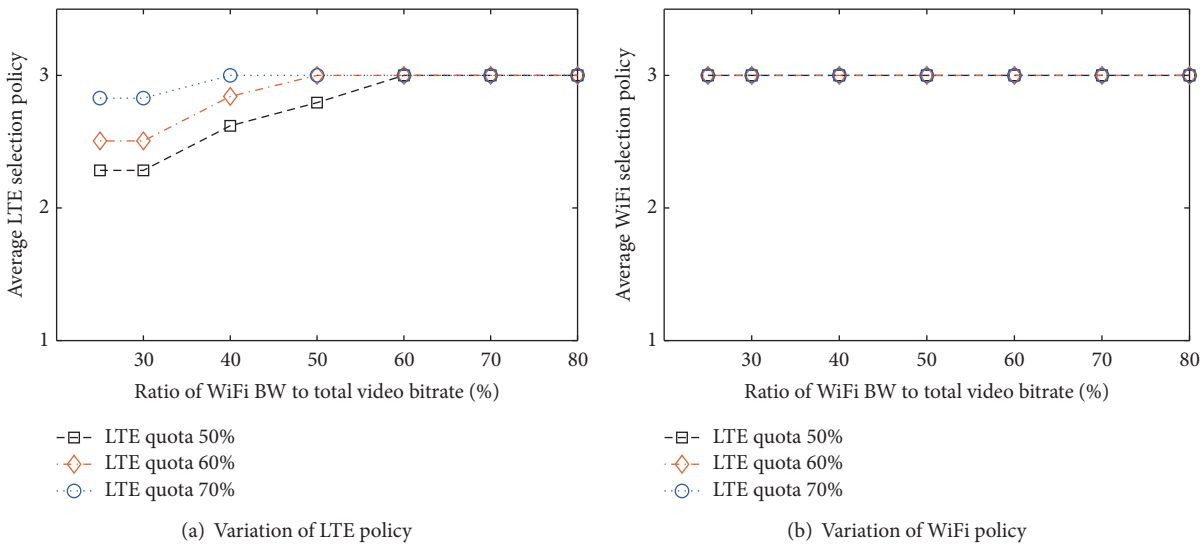
when various LTE data quotas are available. Equation (12) indicates that the LTE data constraint is affected by both WiFi throughput and the user's LTE quota. Thus, as shown in Figure 7(a), the LTE policy is poor when the ratio of WiFi throughput to video bitrate is low. Also, unsurprisingly, when the user's LTE quota is relatively low, the LTE selection policy is also low.

By contrast, as shown in Figure 7(b), the WiFi selection policy is at the highest level, regardless of the amounts of LTE quota and WiFi bandwidth. Equation (12), which does not include any term for WiFi status (X^n_{WiFi}), may explain this phenomenon.

To compare the performances obtained with different amounts of LTE data quota, we measure the LTE data consumption, battery usage, and deadline miss duration. Figure 8(a) shows the results of LTE data consumption in different scenarios over the emulated WiFi environment, where the interface selection manager freely allocates video segments to LTE links when there is a sufficient amount of LTE quota.

As shown in Figure 8(b), power consumption shows a similar pattern to that of the LTE data consumption. Thus, we conclude that battery usage decreases within smooth WiFi environments where less LTE data consumption takes place.

However, the restricted use of LTE data causes deadline misses in the enhancement layers. Figure 8(c) shows the deadline miss duration during 10 min of playback. As depicted in the figure, having 50% of LTE quota remaining leads to the worst QoS, with approximately 8 min of deadline misses in layer 3 over a WiFi bandwidth that is 30% of the total bitrate. Moreover, our proposed policy optimizer prevents reckless use of LTE data even if the device is connected to poor WiFi bandwidths.



(a) Variation of LTE policy

(b) Variation of WiFi policy

FIGURE 7: Optimized (a) LTE and (b) WiFi selection policies when the user has limited LTE data quota (50, 60, and 70% of the total size of the video bitstream).

7.2. Performance in Real-World Environment. In order to evaluate our proposed system in the real-world environment, two real-world scenarios were designed as shown in Figure 9. In these experiments, the available bandwidth fluctuated as the user moved around. First, in mobile scenario #1, the mobile user moves outside of the original WiFi router's coverage, in which a relatively high-download bandwidth was available, so that the device connects to a different WiFi router that offers insufficient bandwidth for approximately 80 s, at the end of which time the mobile device is again positioned within the range of the router offering smooth WiFi coverage.

Then, in mobile scenario #2, the mobile user moves in and out of the original router's coverage for a few seconds at a time. The same video source and contents of the previously discussed optimization experiment are used; there are three layers and the video lasts 120 s.

7.2.1. Trace-Driven Analysis. We analyzed how multi-interface selection system reacts to network fluctuations during device mobility. Figure 10 shows the traces of the aggregated bandwidth and the selected interfaces of the proposed optimal multi-interface selection system during two minutes of video playback in both mobile scenarios.

In Figure 10, the trace-driven numerical result shows that the aggregated throughput over WiFi and LTE interfaces is much higher in the buffering state. If the internal buffer of all layers was filled with video segments, the video segment manager would not send HTTP GET request through any of interface link; then the aggregated throughput is zero. The selected interfaces during the streaming in each mobile scenario is depicted in bottom of Figures 10(a) and 10(b). As shown in the figures, WiFi interface is selected for streaming whole layers when a high (higher than total bitrate of all layers) WiFi bandwidth is available. In contrast, the LTE link is heavily used to download interface at the time intervals of lower WiFi-download bandwidth (e.g., 20–100 s in mobile #1 and 10–30 s, 50–60 s, 80–90 s, and 100–110 s in mobile #2).

7.2.2. Comparison with Different Streaming Schemes. These scenario-based experiments were conducted with different streaming schemes in order to compare their performances with that of our proposed system. The streaming schemes used in these experiments are described below:

(i) Two single-interface, video streaming environments that download video segment via WiFi-only and LTE-only, respectively.

(ii) A multi-interface selection scheme—namely, *INSTANT*—that uses the throughput of the last segment on an interface (without making predictions) to allocate the next download interface.

(iii) A streaming system that implements conventional SVC streaming over multiple interfaces; most previous studies have adopted [6, 27, 28]. The conventional SVC streaming differs from our optimal multi-interface selection system in our system employment of a layer splitting and a policy optimizer.

(iv) A proposed optimal multi-interface streaming scheme that restricts the user's LTE data quota and battery life (e.g., 1% of battery life and an LTE data quota, that is, 70% of the video's total size).

Figure 11(a) shows the deadline miss duration in all six of the different streaming schemes. The proposed optimal multi-interface (*Optimal MI*) and *LTE-only* schemes do not incur deadline misses in either of the two scenarios; every other scheme does. In particular, the *WiFi-only* scheme has the worst QoS, with 66.28 and 15.01 s of deadline misses in scenarios #1 and #2, respectively. Figure 11(b) illustrates the start-up latency of each scheme for both scenarios. The *WiFi-only* and *conventional* schemes have the worst start-up latency

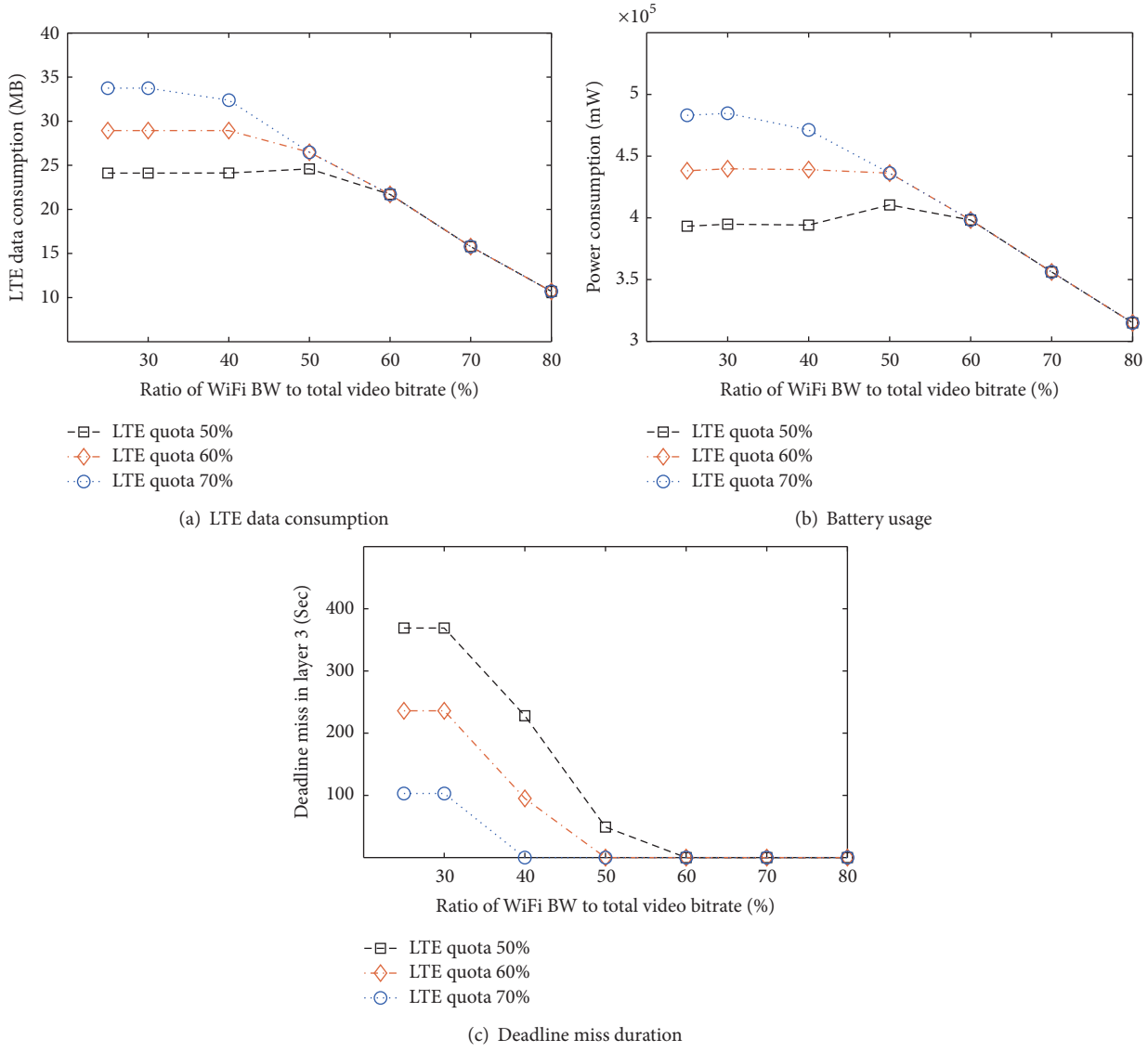


FIGURE 8: Varying (a) LTE data consumption, (b) battery usage, and (c) deadline miss duration according to the ratio of the WiFi bandwidth to the total bitrate of all layers when the user's LTE data quota is limited.

performances because neither of them can use LTE links simultaneously, resulting in longer initial buffering times.

Figures 11(c) and 11(d) show six schemes' LTE data and battery consumption for two min of playback. Even though the *WiFi-only* scheme is unacceptable due to the long duration of its deadline misses, the scheme performs well in LTE data consumption since it does not consume any LTE data. On the other hand, while *LTE-only* plays without any deadline misses, it consumes the highest amount of LTE data; in terms of data-cost efficiency, this is unacceptable. The *INSTANT* scheme consumes the second-most LTE data and power, leading us to conclude that a careless use of multiple network interfaces can result in large data and energy costs.

Our proposed *Optimal MI* scheme, however, provides the highest-video quality without any deadline misses and with LTE consumption that is 47.9–52.6% that of the *LTE-only* scheme and 71.8–83.9% that of the *conventional* scheme. Additionally, regarding battery usage, the *Optimal MI* scheme consumes 72.9–74.3% of the energy consumed by the *LTE-only* scheme and 80.6–96.2% of that consumed by the *conventional* scheme.

Furthermore, the *Optimal MI* scheme uses the policy optimizer, which makes it possible to avoid excessive data and power consumption, as shown in the result of Batt = 1% and Data = 70% configuration.

8. Related Work

In mobile environments, video content providers must be able to guarantee the QoS of their video streaming in unpredictable and unstable heterogeneous wireless interfaces. Thus, *bandwidth aggregation* has emerged as a popular topic

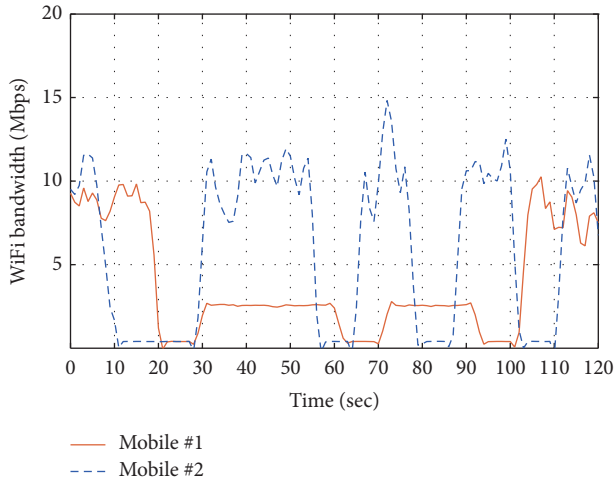
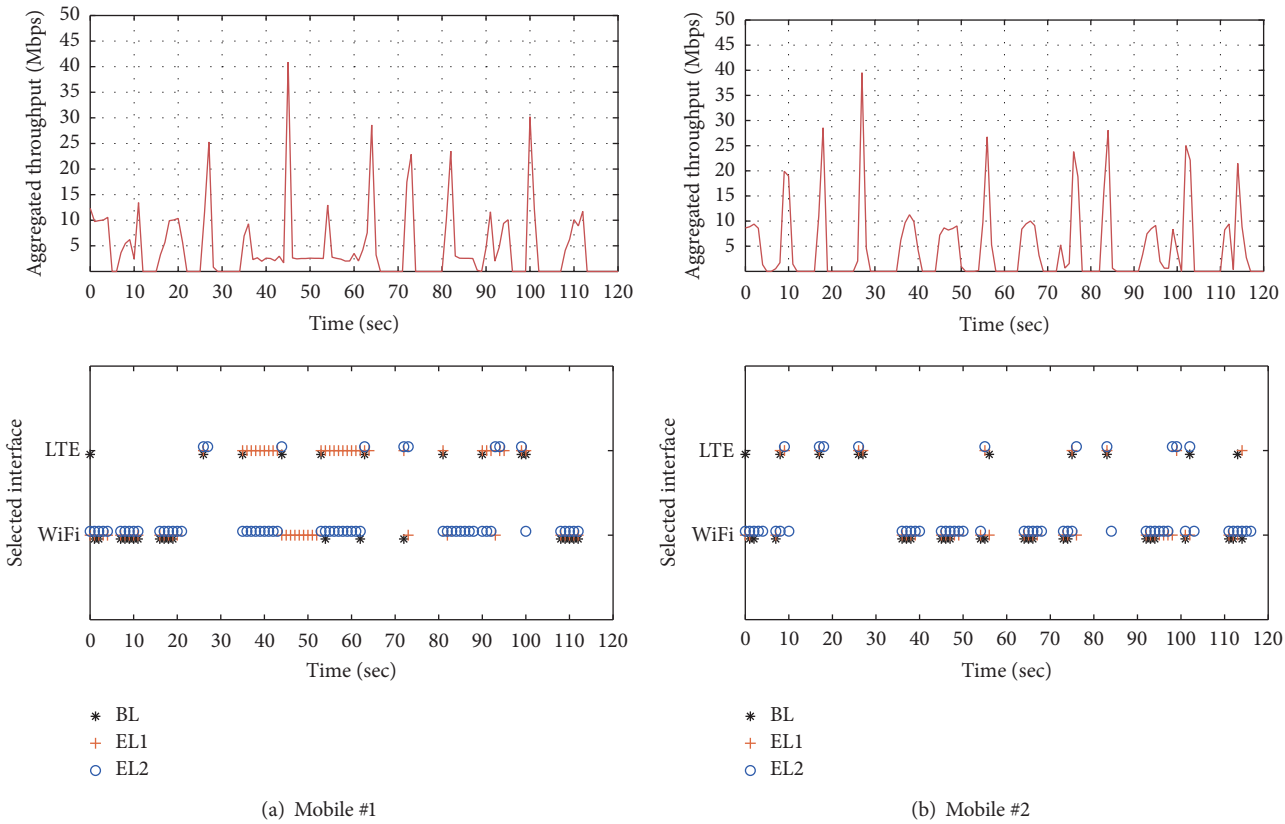


Figure 9: Characterization of real-world mobile environments.



(a) Mobile #1 (b) Mobile #2

Figure 10: The traces of the aggregated bandwidth and the selected interface of the proposed optimal multi-interface selection system for two minutes of video playback in (a) mobile #1 and (b) mobile #2 scenario.

of research regarding multipath network environments. Most previous studies can be divided into two approaches to bandwidth aggregation: those that propose packet scheduling over multipath networks [4–6, 27, 29] and those that propose extending TCP/UDP for multipath (MPTCP) support [7–9].

Chebrolu and Rao [4] have proposed the earliest delivery path first (EDPF) scheduling algorithm, which ensures that packets meet their playback deadlines by scheduling packets based on the estimated delivery time of the packets in each path. To schedule a packet delivery for the shortest path using EDPF, the proxy between the streaming server and mobile host is used to estimate the overall path characteristics (e.g., delay and packet loss). Jurca et al. [5] have also presented an efficient, low-complexity multipath streaming algorithm that may be used for both stored and live video services. They proposed a heuristic-based solution that allows fast assignment

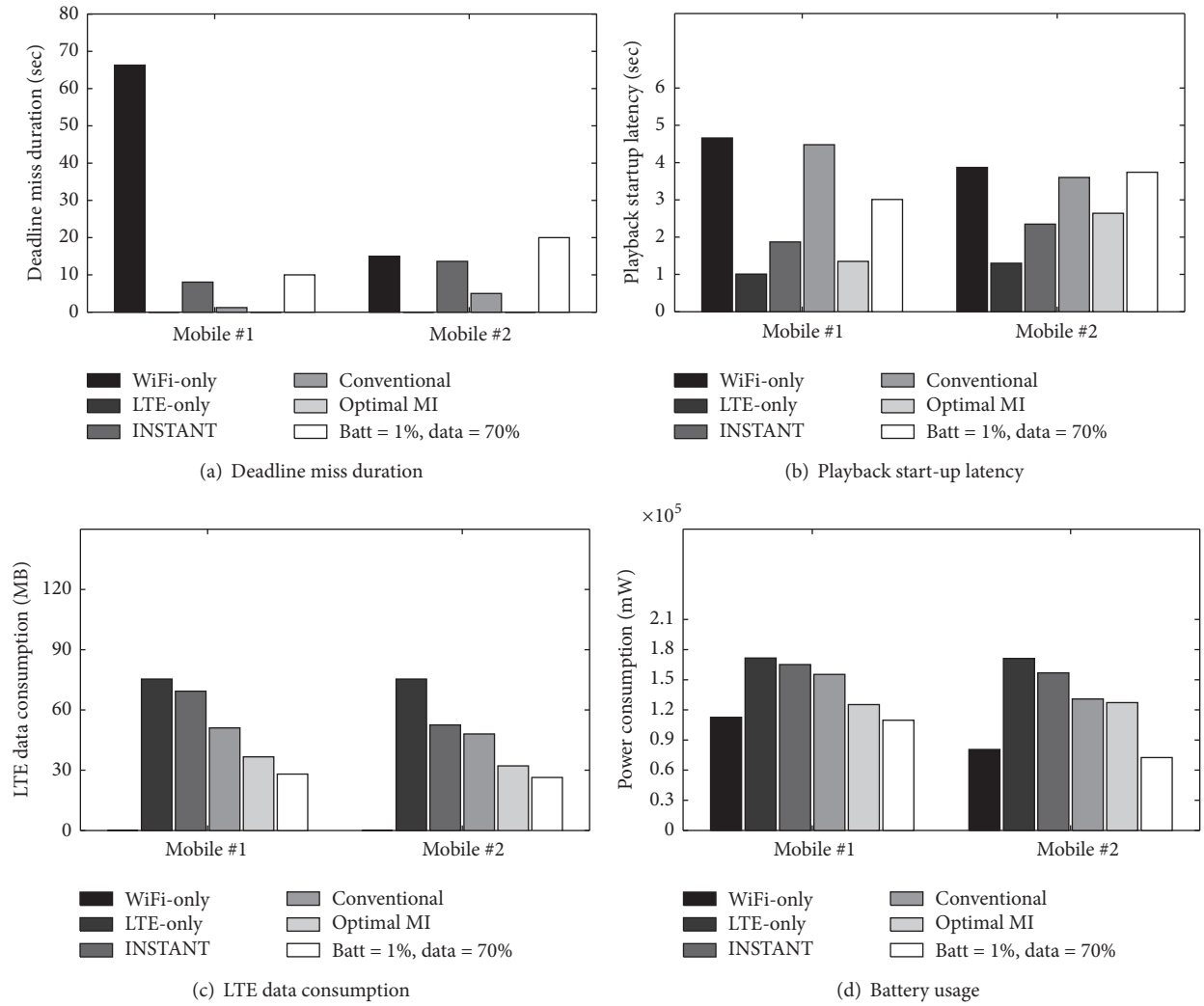


Figure 11: Comparison result between the proposed optimal multi-interface selection system and various streaming system in the real-world environments.

of a packet to the most suitable path during real-time streaming; from the work, the authors derived an optimization problem assuming that the server either knows or can accurately predict the state of the network. Fernandez et al. [6] have proposed a system that allows users to dynamically negotiate QoS profiles with multiple interfaces; they have also presented an enhanced version of the EDPF scheduling algorithm to their own bandwidth allocation scheme.

Our work differs from the above-mentioned studies in the crucial aspect that we aim to provide a client-side request scheduler over a multiple interface environment such as 4G LTE and WiFi. By using client-side request scheduling, the streaming server can avoid both bottlenecks and packet scheduling overhead for each connected client.

Similarly, the authors of [27] have presented a client-side request scheduler that distributes requests for the video over both a public WLAN and a 3G (HSDPA) network simultaneously. They have proposed quality adaptation and HTTP-based request schedulers which are responsible for distributing requests and adjusting the desired video quality in combination with dynamic subsegment approaches. However, because the authors do not use an H.264 scalable extension, the video service thus achieved is limited such that users cannot immediately see the quality change. Likewise, Bui et al. [29] have developed a bandwidth aggregation middleware that supports real-time data streaming services over LTE and WiFi. In [30], the authors proposed an energy-efficient HTTP adaptive streaming algorithm over heterogeneous wireless networks. The main difference between our work and those presented in [27, 29, 30] is that the authors of those works consider a single HTTP-based video stream, which means that their approaches may create out-of-order packet delivery problems when time-sequential (temporal basis) interface scheduling is used. In addition, our proposed system enables adaptively controlling with awareness of user's LTE data quota during a video playback.

Using the Markov decision process, Xing et al. [28] have proposed a real-time optimizing algorithm for video streaming over multiple wireless access networks. However, those authors only experimented in two unlicensed band wireless

networks (such as WiFi and Bluetooth)—not in a cellular network. Thus, their system is restricted to being practically effective only in mobile environments, due to the narrow coverage provided by WiFi and Bluetooth. Additionally, those authors focused on a lower start-up latency and higher video quality without addressing the problems of energy consumption.

In the work in [31], the authors proposed an adaptive LTE/WiFi network interface activation algorithm with a cost function which enables considering user's battery life, LTE data quota, and expected file transfer completion time. However, their proposed system design targeted for file transfer that is supposed to be utilized only as delay-tolerant data offloading. However, we designed a multi-interface selection system with a real-time adaptive scheme for delay-intolerant video streaming services. Hoque et al. [32] presented an energy-efficient multimedia delivery system, namely, EStreamer, to determine an energy-optimal burst size. Particularly, they focused on a potential to save energy using single wireless network interface such as WiFi, 3G (HSPDA), and LTE. In the work in [33], Lee et al. proposed an optimal scheduling policy for resource-efficient mobile video streaming which minimizes the sum of cellular cost and smartphone energy consumption by adaptively using WiFi or LTE, but not simultaneous use of both LTE and WiFi links.

Extending TCP for multipath (MPTCP) support for wired networks is an active area of research. MPTCP (main flow) runs in multihomed mobile devices to simultaneously deliver TCP packets (subflow) over multiple paths and to pool the entire available bandwidth together. Although MPTCP improves the available throughput [7, 8], unresolved problems caused by congestion controls and out-of-order packet deliveries still exist—particularly when the multipath has asymmetric latency, loss, or bandwidth [34, 35]. To support video streaming with MPTCP, Singh et al. [9] have deployed multipath RTP, which extends RTP to multipath communication by allowing a single RTP stream to be split into multiple subflows. That work sends a small fraction of traffic to continuously monitor the path characteristics; furthermore, to avoid skewing the receiver's packet, a relatively small window size is used for media packets. Even though, MPTCP and MPRTP have struggled to resolve out-of-order packet deliveries, these methods can hardly be free from their inconvenient deployment with compatible streaming infrastructures. Chen et al. [36] proposed eMTCP, an energy-aware MPTCP-based content delivery scheme which offloads between LTE and WiFi with tradeoff-awareness of throughput and energy consumption. Their proposed system is available to operate over MPTCP-enabled infrastructure; however, our proposed system operates on the mobile-client sides without requiring any changes to the existing infrastructure and servers.

9. Conclusions

Contemporary mobile devices utilize more than a single-network interface, and these interfaces have different strengths and weaknesses. The demand for suitable use of multiple network interfaces is constantly increasing—particularly, as users' expectations of stable HD video streaming grow. In this study, we have constructed an optimal multi-interface selection system that aims to conserve the user's remaining LTE data quota and battery life. As has been previously mentioned, the simultaneous use of multiple network interfaces provides a larger download bandwidth as much as the sum of each interface's capacity. However, such use also risks consuming enormous amounts of data and energy when the multiple interfaces are used carelessly. In order to resolve this risk, we have defined an optimization problem for optimal interface selection policies, which is based on BIP; this problem includes constraints for the user's LTE data quota and their device's remaining battery life. Moreover, by adopting split-layer SVC encoding, we are able to both address the well-known problem of out-of-order and skewed packet and immediately adjust the video quality to accommodate fluctuating aggregated bandwidth. The experimental comparison results between our proposed system and the conventional mobile video streaming schemes prove the performance of our proposed scheme, which provides the highest-video quality without the excessive consumption of LTE data and battery life.

Important Notations

$\vec{L}$: Split layer of the SVC bitstream $\langle 1, \ldots, n \rangle$
S_n, Q_n: Segment size, video bitrate of layer n
M_n: Total number of segments of layer n
$\vec{V}_{L,i}$: ith video segment in layer L
$\vec{T}_{\text{Net},i}$: Expected throughput of network interface Net
$\vec{A}_{\text{Net},i}$: Available bandwidth of network interface Net
$\overrightarrow{BW}_{\text{Net},i}$: Measured bandwidth of network interface Net
$\vec{t}_{i,L}$: Downloading time of ith segment in layer L
r: Weight for interface decision criteria
w: Weight for smoothness of historical prediction
B_{LTE}: Cumulative amount of data consumption of LTE
$\vec{P}_{\text{Net}}$: Optimized selection policy of network interface Net
$\overrightarrow{\text{SI}}_{L,i}$: Selected interface of ith video segment in layer L.

Competing Interests

The authors declare that they have no competing interests.

Acknowledgments

This research was supported by the MSIP (Ministry of Science, ICT and Future Planning), Korea, under the "ICT Consilience Creative Program" (IITP-R0346-16-1008) supervised by the IITP (Institute for Information & Communications Technology Promotion).

References

[1] KPCB, Internet Trend 2014, http://www.kpcb.com/internet-trends.

[2] Sandvine, "Global internet phenomena report," 2014, https://www.sandvine.com/trends/global-internet-phenomena/.

[3] 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 *Proceedings of the 10th International Conference on Mobile Systems, Applications, and Services (MobiSys '12)*, pp. 225–238, ACM, June 2012.

[4] 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.

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

[6] J. C. 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.

[7] S. Barré, C. Paasch, and O. Bonaventure, "Multipath TCP: from theory to practice," in *NETWORKING 2011*, J. Domingo-Pascual, P. Manzoni, S. Palazzo, A. Pont, and C. Scoglio, Eds., vol. 6640 of *Lecture Notes in Computer Science*, pp. 444–457, Springer, Berlin, Germany, 2011.

[8] D. Wischik, C. Raiciu, A. Greenhalgh, and M. Handley, "Design, implementation and evaluation of congestion control for multipath TCP," in *Proceedings of the 8th USENIX Conference on Networked Systems Design and Implementation (NSDI '11)*, vol. 11, Boston, Mass, USA, March 2011.

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

[10] H. Schwarz, D. Marpe, and T. Wiegand, "Overview of the scalable video coding extension of the H.264/AVC standard," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 17, no. 9, pp. 1103–1120, 2007.

[11] S. Moon, C. Jung, J.-S. Lee, and S. Kim, "On the impact of layer-splitting for cloud-based SVC streaming," in *Proceedings of the 2nd International Conference on Future Internet of Things and Cloud (FiCloud '14)*, pp. 210–215, Barcelona, Spain, August 2014.

[12] Q. He, C. Dovrolis, and M. Ammar, "On the predictability of large transfer TCP throughput," *Computer Networks*, vol. 51, no. 14, pp. 3959–3977, 2005.

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

[14] S. Moon, J. Yoo, and S. Kim, "Exploiting adaptive multi-interface selection to improve qos and cost-efficiency of mobile video streaming," in *Proceedings of the IEEE International Conference on Mobile Services*, pp. 134–141, 2015.

[15] OpenSignal, "The State of LTE 2015," http://opensignal.com/reports/2015/09/state-of-lte-q3-2015/.

[16] L. A. Wolsey, *Integer Programming*, vol. 42, John Wiley & Sons, New York, NY, USA, 1998.

[17] X. Li, M. Dong, Z. Ma, and F. C. A. Fernandes, "GreenTube: power optimization for mobile videostreaming via dynamic cache management," in *Proceedings of the 20th ACM International Conference on Multimedia (MM '12)*, pp. 279–288, ACM, Nara, Japan, November 2012.

[18] M. A. Hoque, M. Siekkinen, J. K. Nurminen, M. Aalto, and S. Tarkoma, "Mobile multimedia streaming techniques: QoE and energy saving perspective," *Pervasive and Mobile Computing*, vol. 16, pp. 96–114, 2015.

[19] Y. Liu and M. Wang, "An application-layer approach for energy-efficient multimedia streaming," in *Proceedings of the International Conference on Computing, Networking and Communications (ICNC '15)*, pp. 742–748, IEEE, Garden Grove, Calif, USA, February 2015.

[20] P. International, Kill a watt control, http://www.p3international.com/products/p4482.html.

[21] Apple, Introducing swift, https://developer.apple.com/swift/.

[22] R. Hanson, "Asynchronous socket networking library for mac and ios," https://github.com/robbiehanson/CocoaAsyncSocket.

[23] Apple, "Get battery status," https://developer.apple.com/library/ios/samplecode/BatteryStatus/Introduction/Intro.html.

[24] J. Clausen, *Branch and Bound Algorithms-principles and Examples*, Department of Computer Science, University of Copenhagen, 1999.

[25] A. Kivity, Y. Kamay, D. Laor, U. Lublin, and A. Liguori, "KVM: the linux virtual machine monitor," in *Proceedings of the Linux Symposium*, vol. 1, Ottawa, Canada, June 2007.

[26] I. Joyent, About node.js, http://nodejs.org/about/.

[27] K. Evensen, D. Kaspar, C. Griwodz, P. Halvorsen, A. Hansen, and P. Engelstad, "Improving the performance of quality-adaptive video streaming over multiple heterogeneous access networks," in *Proceedings of the 2nd Annual ACM Conference on Multimedia Systems (MMSys '11)*, pp. 57–68, ACM, Santa Clara, Calif, USA, February 2011.

[28] M. Xing, S. Xiang, and L. Cai, "A real-time adaptive algorithm for video streaming over multiple wireless access networks," *IEEE Journal on Selected Areas in Communications*, vol. 32, no. 4, pp. 795–805, 2014.

[29] D. H. Bui, K. Lee, S. Oh et al., "GreenBag: energy-efficient bandwidth aggregation for real-time streaming in heterogeneous mobile wireless networks," in *Proceedings of the IEEE 34th Real-Time Systems Symposium (RTSS '13)*, pp. 57–67, IEEE, Vancouver, Canada, December 2013.

[30] Y. Go, O. C. Kwon, and H. Song, "An energy-efficient HTTP adaptive video streaming with networking cost constraint over heterogeneous wireless networks," *IEEE Transactions on Multimedia*, vol. 17, no. 9, pp. 1646–1657, 2015.

[31] W. Lee, J. Koo, S. Choi, and Y. Park, "E$PA: energy, usage ($), and performance-aware LTE-WiFi adaptive activation scheme for smartphones," in *Proceedings of the IEEE 15th International Symposium on a World of Wireless, Mobile and Multimedia Networks (WoWMoM '14)*, pp. 1–9, Sydney, Australia, June 2014.

[32] M. A. Hoque, M. Siekkinen, J. K. Nurminen, S. Tarkoma, and M. Aalto, "Saving energy in mobile devices for on-demand multimedia streaming—a cross-layer approach," *ACM Transactions on Multimedia Computing, Communications, and Applications (TOMM)*, vol. 10, no. 3, article 25, 2014.

[33] J. Lee, K. Lee, C. Han, T. Kim, and S. Chong, "Resource-efficient mobile multimedia streaming with adaptive network selection," *IEEE Transactions on Multimedia*, vol. 18, no. 12, pp. 2517–2527, 2016.

[34] C. Raiciu, C. Paasch, S. Barre et al., "How hard can it be? designing and implementing a deployable multipath TCP," in *Proceedings of the 9th USENIX conference on Networked Systems Design and Implementation*, pp. 29–29, USENIX Association, San Jose, Calif, USA, 2012.

[35] Y.-C. Chen, Y.-S. Lim, R. J. Gibbens, E. M. Nahum, R. Khalili, and D. Towsley, "A measurement-based study of multipath TCP performance over wireless networks," in *Proceedings of the Conference on Internet Measurement Conference (IMC '13)*, pp. 455–468, ACM, 2013.

[36] S. Chen, Z. Yuan, and G.-M. Muntean, "An energy-aware multipath-TCP-based content delivery scheme in heterogeneous wireless networks," in *Proceedings of the IEEE Wireless Communications and Networking Conference (WCNC '13)*, pp. 1291–1296, Shanghai, China, April 2013.