# Reliable and Energy-Efficient Hybrid Screen Mirroring Multicast System

Yunmin Go and Hwangjun Song

**Abstract—**This paper presents a reliable and energy-efficient hybrid screen mirroring multicast system for sharing high-quality real-time multimedia service with adjacent mobile devices over WiFi network. The proposed system employs overhearing-based multicast transmission scheme with Raptor codes and NACK-based retransmission to overcome well-known WiFi multicast problems such as low transmission rate and high packet loss rate. Furthermore, to save energy on mobile devices, the proposed system not only shapes the screen mirroring traffic, but also determines the target sink device and Raptor encoding parameters such as the number of source symbols, symbol size, and code rate while considering the energy consumption and processing delay of the Raptor encoding and decoding processes. The proposed system is fully implemented in Linux-based single board computers and examined in real WiFi network. Compared to existing systems, the proposed system can achieve good energy efficiency while providing a high-quality screen mirroring service.

**Index Terms—**Screen content, screen mirroring, WiFi, multicast, systematic raptor codes, overhearing

## 1 Introduction

Screen mirroring technology enables a mobile device to duplicate its screen content in real-time onto a large display device, such as monitor, TV, and projector. This technology allows the mobile user to overcome the constraints of the small display unit in a mobile device. Furthermore, screen mirroring can be applicable to various applications, such as gallery sharing, presentations, mobile streaming, and mobile gaming [1], [2], [3], [4], [5]. Because of its wide range of applications, state-of-the-art mobile devices typically offer screen mirroring functionality, and some commercial products are already available, e.g., AirPlay [6], Chromecast [7], MirrorOp [8], Splashtop [9], and Miracast [10]. In particular, Miracast, which is developed by the WiFi Alliance, aims to act like a wireless High Definition Multimedia Interface (HDMI) cable. In Miracast, the source device (i.e., the mobile device) encodes the screen content with H.264/AVC and transmits the compressed video data to the sink device (i.e., typically WiFi-enabled receiver connected to a TV or display device) using Real-Time Streaming Protocol (RTSP) and WiFi-Direct. Recently, the demand for screen content sharing among adjacent mobile devices has been increasing for conferences, lectures, etc. However, it is still challenging to provide screen mirroring for multiple adjacent devices because existing screen mirroring technologies support only one-to-one connection.

To handle this problem, it is necessary to enable WiFi multicast for screen mirroring. Unfortunately, there are several well-known problems in the WiFi multicast. One of the most serious problems is unreliable packet delivery caused by the absence of acknowledgment and packet retransmission request. Another problem is that the sender selects a low transmission rate and high transmission power level to deliver the data even to the farthest receiver from the sender. Therefore, the existing WiFi multicast is not suitable to provide a high-quality screen mirroring service, which requires high video bitrate and error robustness. To solve this problem and provide multicast video streaming over WiFi network, some research efforts [11], [12] have been devoted to overhearing and forward error correction (FEC)-based multicast transmission. In this method, the sender delivers the data to the target receiver using unicast transmission while the non-target receivers overhear the unicast transmission. Because the rate adaptation and MAC-layer retransmission are operated by the unicast transmission between the sender and the target receiver, high transmission rate can be achieved. Moreover, FEC schemes are employed to provide reliable data delivery to the non-target receivers who cannot utilize the MAC-layer retransmission. Recently, some fountain codes-based video streaming methods have been proposed to provide error-resilient multimedia services in the literature [13], [14], [15], [16], [17]. The fountain codes have been successfully deployed for streaming applications, due to their flexibility, coding efficiency, and computational complexity [18].

However, it is still difficult to utilize overhearing and fountain codes for screen mirroring multicast in the state-of-the-art mobile devices because of their limited battery capacity and computing power. It is well-known that a WiFi network interface can consume approximately three times the energy required to decode audio or video content [19], [20]. The main reason is that the wireless network interface of the mobile device maintains the active state to receive continuous data during the streaming service. However,

- *Y. Go is with the Division of IT Convergence Engineering, Pohang University of Science and Technology, Pohang-si 790-784, Republic of Korea. E-mail: gnfservant@postech.ac.kr.*
- *H. Song is with the Department of Computer Science and Engineering, Pohang University of Science and Technology, Pohang-si 790-784, Republic of Korea. E-mail: hwangjun@postech.ac.kr.*

*Manuscript received 2 Nov. 2016; revised 16 June 2017; accepted 23 June 2017. Date of publication 0 . 0000; date of current version 0 . 0000.*
*(Corresponding author: Hwangjun Song.)*
*Recommended for acceptance by C.F. Chiasserini.*
*For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.*
*Digital Object Identifier no. 10.1109/TMC.2017.2721947*

energy efficiency can be improved by shaping multimedia traffic into periodic burst patterns [20], [21]. A number of packets are collected at a time and sent together to a receiver through a wireless network. In this case, the wireless network interface at the receiver remains in the active state for only a short period, instead of staying in the active state consistently. Furthermore, when fountain codes are adopted, the mobile device should perform supplementary fountain encoding and decoding processes which require additional energy and delay. In fact, the amounts of energy and delay required for fountain encoding and decoding processes change significantly according to the encoding parameters of the fountain codes (e.g., code rate, symbol size, and the number of source symbols) [36].

In this paper, we propose a reliable and energy-efficient hybrid screen mirroring multicast system for sharing high-quality screen content among adjacent mobile devices. In the proposed system, the overhearing-based multicast scheme is employed to overcome well-known problems of the WiFi multicast. To mitigate the video quality degradation caused by packet loss, the proposed system utilizes systematic Raptor codes [22] as an FEC scheme and NACK-based retransmission scheme as an ARQ scheme for error correction. Raptor codes are a class of fountain codes [13] and a block-based FEC scheme that provide systematic coding, flexibility, coding efficiency, and rateless codes. These characteristics are very useful for transmitting delay-sensitive data over error-prone wireless networks. The proposed system is designed to minimize energy consumption at the source device and sink devices while still providing a high-quality screen mirroring service. To achieve this goal, an energy consumption model of a WiFi network interface is derived, and then simple but effective energy consumption and delay models for Raptor encoding and decoding processes are obtained. Based on the derived models, the proposed system is designed to shape the screen mirroring traffic based on the buffer occupancy of the sink device and determine the target sink device and Raptor encoding parameters to minimize the overall energy consumption. Our main contributions are summarized below:

- Introduction of an energy consumption model of a WiFi network interface for the overhearing-based multicast scheme.
- Introduction of energy consumption and delay models for the SIMD-based Raptor encoding and decoding processes.
- Design of a target sink device and code rate determining algorithm for the overhearing-based multicast environment by taking into account the wireless network conditions and the energy consumption of WiFi network interfaces.
- Adjustment of Raptor encoding parameters such as code rate, symbol size, number of source symbols, and number of Raptor encoding blocks on the fly by considering time-varying wireless networks and energy consumption of Raptor encoding and decoding processes.
- Implementation of the entire proposed system on Linux-based single board computers and examination of the proposed system in real wireless network environments.
- Achieving spatial video quality improvement of 4.37 dB compared to DirCast [12] and energy savings of 39.05 percent compared to the ACK-based multicast [44] while providing the same level of video quality.

The rest of this paper is organized as follows. In Section 2, we introduce related studies. Details of the proposed system are presented in Section 3. Experimental results are provided in Section 4, and concluding remarks are given in Section 5.

## 2 Related Work

So far, many research efforts have been devoted to screen mirroring [2], [23], [24], [25], [26], [27]. Hsu et al. [2] compared the performance of state-of-the-art screen mirroring technologies. According to their measurements, there is no single winning screen mirroring technology, and there is some room for improvement through design considerations, such as rate adaptation mechanisms and error resilience tools. Furthermore, they implemented a rate adaptation mechanism for a screen mirroring platform in [23]. Chandra et al. [24] presented practical screen sharing system in resource constrained environment. They developed a simple mechanism to transform inter-update temporal redundancy into intra-update spatial redundancy, and achieved good compression rates and high screen capture rates. Zhang et al. [25] conducted a measurement study on the power consumption of Miracast. Using insights from the measurement, they proposed some energy efficient mechanisms such as adaptive video tail cutting, redundant codec operation bypass, and least congested channel selection. In [26], Ha et al. presented a frame filtering method that reduces the Miracast traffic load by analyzing the dynamism of screen content. Similarly, Bae et al. [27] proposed an adaptive frame skipping method that analyzes the motion dynamics of screen content. However, it is still challenging to provide screen mirroring multicast because existing screen mirroring systems are limited to unicast transmission.

To solve this problem, it is necessary to enable WiFi multicast for screen mirroring. To date, many research efforts have focused on providing multicast media delivery system over WiFi network [11], [12], [16], [17], [28], [29], [30], [31]. Choi et al. [28] proposed a leader-based multicast service (LBMS) to improve the reliability and efficiency of WiFi multicast. Although the leader client of a multicast group can send feedback frames for retransmission request and rate adaptation, LBMS still cannot provide sufficient goodput for high-quality video multicast. To improve the goodput of a WiFi multicast, Park et al. [29] used the unicast transmission to deliver the IPTV stream to a target receiver, while non-target receivers overhear the unicast transmission. This WiFi multicast transmission method is called pseudo-broadcast. Their analysis show that the pseudo-broadcast achieves high transmission rate with retransmission and rate adaptation for only one target receiver, and it cannot provide reliable packet delivery for non-target receivers. In [11], Sen et al. proposed a pseudo-broadcast based WiFi system for high-quality media delivery called Medusa. Medusa first estimates the priority of a packet according to the dependency of video decoding. Based on the priority of the packet, it performs PHY rate adaptation, packet order selection, and network coded retransmission. Similar to Medusa, Chandra et al. [12] proposed DirCast to minimize the airtime consumed by multicast traffic, DirCast determines the destination client for pseudo-broadcast and assigns an appropriate multicast group for a joining client. In addition, DirCast uses a proactive and adaptive FEC scheme to reduce the loss rate. Lin et al. [16] presented enhanced random early detection (ERED)-FEC mechanism at WiFi access point (AP). With the wireless channel condition and network traffic load information, ERED-FEC calculates the FEC redundancy rate to improve the video quality without overloading the network. In [17], energy-efficient and content-aware FEC mechanism is proposed, which minimizes the overall transmission rate while satisfying the perceptual video quality requirement. There also have been research efforts to adopt the Raptor codes for WiFi multicast. Chiao et al. [30] demonstrated that the performance of systematic Raptor codes could protect a WiFi multicast streaming system inside a high-speed train. According to their analysis, the systematic Raptor code is an effective approach to recover the lost packets of a WiFi multicast. Choi et al. [31] presented a reliable video multicast scheme based on the Raptor codes and coordination between multiple APs. In the system, each AP transmits entirely or partially different Raptor encoded packets for reliable video multicast, and allocates the resource for Raptor code rate adaptation.

One of the major concerns of screen mirroring multicast technology is the amount of energy needed by the mobile device to conduct multicast transmission over the wireless network. To date, many energy-efficient wireless networking technologies [19], [32], [33], [34], [35], [36] have been proposed to improve the energy efficiency of the network interface on a mobile device with limited battery capacity. In [32], it was shown that traffic shaping using a proxy server can save more energy than adjusting the mobile device sleep time. Hoque et al. [19] presented an EStreamer to provide energy-efficient multimedia streaming service. EStreamer determines an optimal burst size and idle period length for a streaming client to allow the mobile device to reduce their energy consumption with seamless multimedia streaming. Shen et al. [33] proposed a Gaussian mixture model to reflect the video watching time of users over the Internet and developed a traffic shaping algorithm to determine the optimal buffering points during video playback based on the Gaussian mixture model. In [34], Poellabauer et al. proposed an effective approach to increase the time interval for a network interface to remain in doze mode and active mode. Go et al. [35] proposed a seamless high-quality HTTP adaptive streaming algorithm that considers wireless network conditions and the energy consumption of a mobile device with networking cost constraints over heterogeneous wireless networks. Kwon et al. [36] proposed a systematic Raptor codes-based energy-efficient multipath streaming transport protocol to support a seamless high-quality video streaming over heterogeneous wireless networks.

## 3 Proposed Screen Mirroring Multicast System

The proposed system aims to provide a high-quality and energy-efficient screen mirroring multicast service among adjacent mobile devices over WiFi network. As mentioned earlier, systematic Raptor codes [22] and NACK-based retransmission scheme are adopted in the proposed system to recover lost packets over the error prone WiFi network. For energy saving at mobile devices, the proposed system shapes the screen mirroring traffic, and determines the target sink device and the Raptor encoding parameters based on the estimated energy consumption models of the WiFi

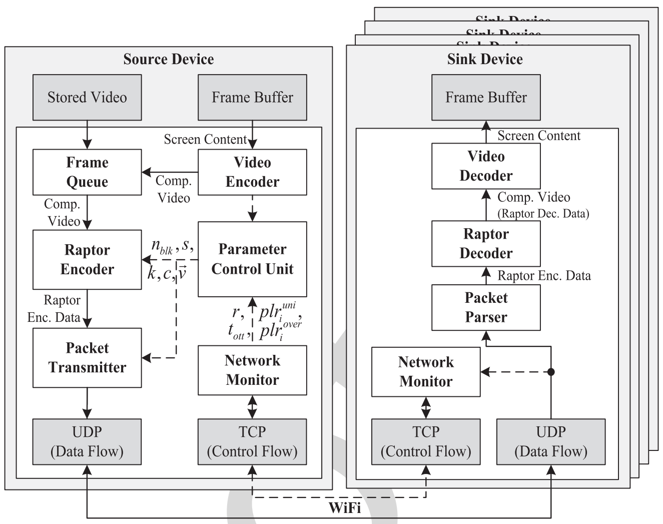

Fig. 1. Overall architecture of the proposed system.

network interface and Raptor encoding and decoding processes. The overall architecture of the proposed system is illustrated in Fig. 1.

As shown in Fig. 1, the proposed system is implemented on both the source device and sink devices. When the mirroring content is an encoded video file stored on the source device, the compressed video data are directly delivered to the frame queue. When the mirroring content is the current screen content on the source device, the frame buffer data of the source device is moved to the video encoder for compression, and then the compressed video data are transferred to the frame queue. The frame queue pushes the compressed frame data to the Raptor encoder. The Raptor encoder processes the frame data with the Raptor encoding parameters determined by the parameter control unit, and then the Raptor encoding blocks are transferred to the packet transmitter. The Raptor encoding blocks are staying at the packet transmitter for a specific time duration to make the burst stream [20], [21]. Finally, the stream is transmitted to multiple sink devices by the packet transmitter. At this time, the proposed system employs overhearing-based multicast transmission to overcome the limitations of WiFi multicast. For a Raptor encoding block, the source device uses a unicast transmission to a target sink device, while the non-target sink devices overhear the transmitted data. When the sink device receives the Raptor encoding blocks, the network monitor periodically observes the WiFi network conditions including transmission rate, round trip time (RTT), and packet loss rate (PLR). This observed information is sent to the source device using a feedback message through TCP control flow, and is utilized to consider the buffered video playback time and WiFi network conditions of the sink devices. In the WiFi network, the MAC-layer retransmission mechanism is supported for unicast transmission. Owing to the lack of support, non-target sink devices can only overhear the unicast transmission. Thus, the PLR of overhearing is generally larger than that of unicast transmission. However, it is still very difficult to recover lost packets when the unexpected extreme packet losses occur. In this case, the sink device requests retransmission to the source device by transmitting NACK message with information of lost packets via TCP control flow.

Examples of traffic shaping are presented in Fig. 2. In the proposed system, a bin duration $t_{bin}$ is defined as the basic

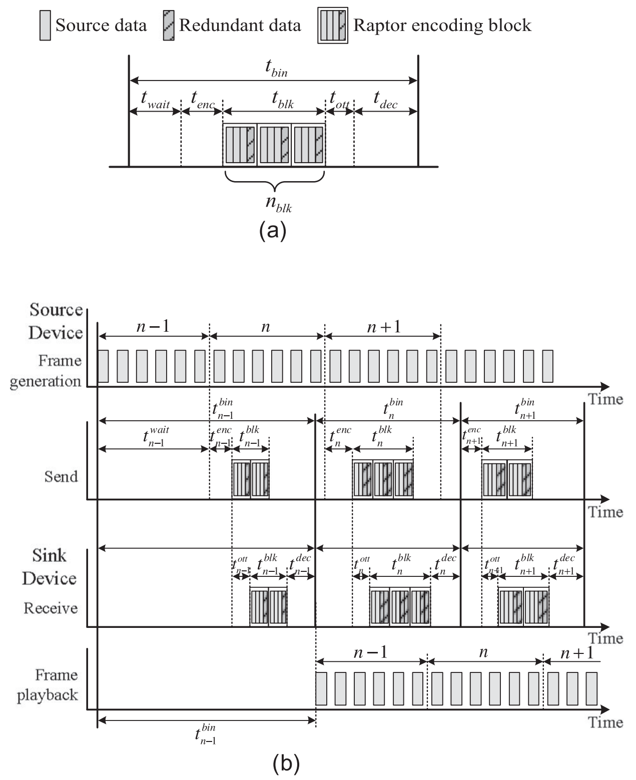

Fig. 2. Examples of traffic shaping: (a) Traffic shaping in a bin duration. (b) Simple example of the traffic shaping in the source device and sink device.

interval for traffic shaping, which includes the video frame waiting time $t_{wait}$, the Raptor encoding delay $t_{enc}$, the transmission delay of Raptor encoding blocks $t_{blk}$, the one-way trip time between the source device and sink device $t_{ott}$ (i.e., half of the maximum RTT value reported from all sink devices), and the Raptor decoding delay $t_{dec}$. That is, $t_{bin}$ is calculated by

$$t_{bin} = t_{wait} + t_{enc} + t_{blk} + t_{ott} + t_{dec}. \tag{1}$$

The frame queue of the source device waits for $t_{wait}$ until a sufficient number of video frames (greater than the size of $n_{blk}$ Raptor encoding blocks) are generated. As soon as a sufficient number of video frames arrive, they are encoded into $n_{blk}$ Raptor encoding blocks with the same Raptor encoding parameters (symbol size $s$, the number of source symbols $k$, and code rate $c$), and then the Raptor encoding blocks are transmitted to the sink devices. The Raptor decoding process is triggered when the corresponding Raptor encoding block arrives at the sink device. In this paper, $c$ is calculated using $c = k/n$, where $n$ is the number of encoding symbols [36], [37]. Actually, $t_{bin}$ is determined by the amount of pure video data transmitted during $t_{blk}$. As $t_{bin}$ increases, the playback delay increases while energy-efficiency is improved because the WiFi network interface of the sink device can stay in the inactivate state longer, and vice versa.

## 3.1 Problem Description

In this section, we formulate the optimal problem; before presenting a detailed description, some key symbols are listed in Table 1. Target sink device selection vector $\vec{v}$ is represented by

TABLE 1
Description of Key Symbols

| Symbol | Description |
|---|---|
| $\vec{v}$ | Target sink device selection vector |
| $s$ | Symbol size of Raptor encoding block |
| $k$ | Number of source symbols for Raptor encoding block |
| $c$ | Coder rate of Raptor encoding block |
| $n_{blk}$ | Number of Raptor encoding blocks |
| $n_{pkt}^{src}$ | Number of source data (video data) packets in a Raptor encoding block |
| $e_{net}(\vec{v}, n_{blk}, s, k, c)$ | Total energy consumption of the WiFi network interfaces at the source and sink devices during $t_{bin}$ |
| $e_{rap}(\vec{v}, n_{blk}, s, k, c)$ | Total energy consumption for Raptor encoding and decoding processes during $t_{bin}$ |
| $t_{play}(\vec{v}, n_{blk}, s, k, c)$ | Playback delay between source and sink device |
| $\tilde{e}_{src}(n_{blk}, s, k)$ | Energy consumption of the WiFi network interface at the source device |
| $\tilde{e}_{i}^{sink}(n_{blk}, s, k)$ | Energy consumption of the WiFi network interface at the $i$-th sink device |
| $\tilde{e}_{enc}(s, k, c)$ | SIMD-based Raptor encoding energy consumption for a Raptor encoding block at the source device |
| $\tilde{e}_{i}^{dec}(s, k)$ | SIMD-based Raptor decoding energy consumption for a Raptor encoding block at the $i$-th sink device |
| $t_{enc}(s, k, c)$ | Raptor encoding delay at the source device |
| $t_{i}^{dec}(s, k)$ | Raptor decoding delay at the $i$-th sink device |
| $E_i(v_i, n_{blk})$ | Expected number of Raptor encoding blocks to be decoded at the $i$-th sink device |
| $\phi_{i}^{dec}(s, k, c)$ | Raptor decoding failure rate for the Raptor encoding block at the $i$-th sink device |
| $N_{sink}$ | Number of sink devices |
| $S_{pkt}$ | Packet payload size |
| $T_{play}^{\max}$ | Tolerable playback delay between the source and sink devices |
| $\Phi_{dec}^{\max}$ | Tolerable maximum Raptor decoding failure rate |

$$\vec{v} = (v_1, v_2, \ldots, v_{N_{sink}})$$
$$v_i = \begin{cases} 1 & \text{if the } i\text{-th sink device selected to the target sink device,} \\ 0 & \text{otherwise.} \end{cases}$$

Energy consumption per packet for WiFi network interfaces and Raptor encoding and decoding processes is calculated by

$$e_{pkt}(\vec{v}, n_{blk}, s, k, c) = \frac{1}{n_{blk} \cdot n_{pkt}^{src}} \cdot \left\{ e_{net}(\vec{v}, n_{blk}, s, k, c) + e_{rap}(\vec{v}, n_{blk}, s, k, c) \right\}, \tag{2}$$

where $n_{pkt}^{src}$ is calculated by $n_{pkt}^{src} = \lceil (s \cdot k)/S_{pkt} \rceil$. The playback delay between source and sink device can be obtained by

$$t_{play}(\vec{v}, n_{blk}, s, k, c) = t_{wait} + n_{blk} \cdot t_{enc}(s, k, c) + t_{blk} + t_{ott} + \max_{1 \le i \le N_{sink}} \left\{ E_i(v_i, n_{blk}) \cdot t_{i}^{dec}(s, k) \right\}, \tag{3}$$

In the proposed system, Raptor decoding is not performed when all original source symbols have successfully arrived at the sink device. Thus, $E_i(v_i, n_{blk})$ is used instead of $n_{blk}$ in Eq. (3) (Please refer to Eq. (23)). Now, we can formulate the problem to achieve our goal as follows.

*Optimal Problem Formulation*: Determine $\vec{v}$, $n_{blk}$, $s$, $k$, and $c$ to minimize

$$\begin{gathered} e_{pkt}(\vec{v}, n_{blk}, s, k, c) \\ \text{subject to } t_{play}(\vec{v}, n_{blk}, s, k, c) < T_{play}^{\max}, \end{gathered} \tag{4}$$

$$\text{and } \max_{1 \leq i \leq N_{sink}} \phi_i^{dec}(s, k, c) \leq \Phi_{dec}^{\max}. \tag{5}$$

Eq. (4) implies that the video frame of the source device should arrive within $T_{play}^{\max}$ in order to avoid buffer underflows at the sink devices, and Eq. (5) indicates that the source device generates sufficient number of redundant packets to support reliable screen mirroring multicast service for all sink devices (Please refer to Eq. (27)).

In fact, the control parameters of the optimal problem formulation (i.e., $\vec{v}$, $n_{blk}$, $s$, $k$, and $c$) are tightly coupled with each other. For example, $\vec{v}$ and $c$ are strongly related to the amount of packets to be transmitted to the sink devices for the successful Raptor decoding. Furthermore, $\vec{v}$ and $c$ depend on both $s$ and $k$ because they are tightly coupled with the robustness of the Raptor encoding block. Hence, it is very difficult to obtain an optimal solution in real-time. In the proposed system, to obtain a feasible solution with a low computational complexity for real-time processing, the above problem is divided into two sub-problems: the target sink device and code rate determining problem, and the Raptor encoding parameter selection problem. The first sub-problem determines $\vec{v}$ and $c$ simultaneously because the channel states of overhearing sink devices strongly depend on which sink device is selected as a target sink device, and the code rate should be adjusted according to the corresponding channel states. Then, the second sub-problem selects Raptor encoding parameters ($n_{blk}$, $s$, and $k$) in a bin duration with the predetermined $\vec{v}$ and $c$. The feasible solution is obtained by performing the two algorithms iteratively.

We first describe the target sink device and code rate determining problem. The proposed system determines the target sink device and code rate to minimize the number of redundant packets for Raptor encoding blocks and the number of retransmitted packets for unexpected packet losses. In fact, the target sink device selection vector $\vec{v}$ and code rate $c$ affect the energy consumption of mobile devices because $\vec{v}$ and $c$ are strongly related to the number of redundant packets and retransmitted packets. The number of redundant packets and retransmitted packets depend on $\vec{v}$ since the number of received packets at each sink device depends on the selected target sink device. Moreover, when $c$ decreases (i.e., the number of redundant packets increases), the number of sink devices which can recover lost packets without retransmission gradually increase, but unnecessary redundant packets can only be received at some sink devices. On the other hand, when $c$ increases (i.e., the number of redundant packets decreases), the amount of retransmitted packets may increase because the number of redundant packets required for successful Raptor decoding may not be sufficient. To achieve our goal, we first define the number of unnecessary redundant packets $n_{pkt}^{unre}(\vec{v}, c)$ as follows.

$$n_{pkt}^{unre}(\vec{v}, c) = \sum_{i=1}^{N_{sink}} \max\left\{n_i^{recv}(v_i, c) - n_{pkt}^{\min}, 0\right\}, \tag{6}$$

$$n_i^{recv}(v_i, c) = n_{pkt}^{blk} \cdot \left[1 - \left\{v_i \cdot plr_i^{uni} + (1 - v_i) \cdot plr_i^{over}\right\}\right], \tag{7}$$

$$n_{pkt}^{\min} = \left\lceil \frac{s \cdot k \cdot (1 + \delta(k))}{S_{pkt}} \right\rceil, \tag{8}$$

$$n_{pkt}^{blk} = \left\lceil \frac{s \cdot k}{c \cdot S_{pkt}} \right\rceil, \tag{9}$$

where $n_i^{recv}(v_i, c)$ is the number of successfully received packets at the $i$th sink device, $n_{pkt}^{\min}$ is the minimum number of packets required for successful Raptor decoding, $\delta(k)$ is the minimum symbol overhead, $n_{pkt}^{blk}$ is the number of packets in a Raptor encoding block, and $plr_i^{uni}$ and $plr_i^{over}$ are the PLR of unicast transmission and the PLR of overhearing at the $i$th sink device, respectively. The number of retransmitted packets $n_{pkt}^{retr}(\vec{v}, c)$ is calculated by

$$n_{pkt}^{retr}(\vec{v}, c) = \sum_{i=1}^{N_{sink}} n_i^{lost}(v_i, c), \tag{10}$$

$$n_i^{lost}(v_i, c) = \begin{cases} n_i^{recv}(v_i, c) & \text{if } n_i^{recv}(v_i, c) < n_{pkt}^{\min}, \\ 0 & \text{otherwise,} \end{cases} \tag{11}$$

where $n_i^{lost}(v_i, c)$ is the number of lost packets at the $i$th sink device. Now, we can formulate the target sink device and code rate determining problem to determine $\vec{v}$ and $c$ as follows.

*Sub-Problem Formulation 1. Target Sink Device and Code Rate Determining Problem:* Determine $\vec{v}$ and $c$ to minimize the cost function $n_{pkt}^{cor}(\vec{v}, c)$

$$\begin{gathered} n_{pkt}^{unre}(\vec{v}, c) + n_{pkt}^{retr}(\vec{v}, c) \\ \text{subject to } 0 < c \leq 1, \end{gathered} \tag{12}$$

$$\text{and } \max_{1 \leq i \leq N_{sink}} \phi_i^{dec}(s, k, c) \leq \Phi_{dec}^{\max}. \tag{13}$$

Finally, the above optimal problem formulation can be simplified as follows with the determined $\vec{v}$ and $c$ from the first sub-problem.

*Sub-Problem Formulation 2. Encoding Parameters Selection Problem:* Determine $n_{blk}$, $s$, and $k$ to minimize the cost function $e_{pkt}^{sel}(n_{blk}, s, k)$ with the given $\vec{v}$ and $c$

$$\begin{gathered} \frac{1}{n_{blk} \cdot n_{pkt}^{src}} \cdot \left\{e_{net}(n_{blk}, s, k) + e_{rap}(n_{blk}, s, k)\right\} \\ \text{subject to } t_{play}(\vec{v}, n_{blk}, s, k, c) < T_{play}^{\max}. \end{gathered} \tag{14}$$

Actually, when the target sink device changes, fairness problem in the performance at sink devices may occur. In general, non-target sink devices require more redundant packets and retransmission packets than the target sink device since they only overhear the unicast transmission. In addition, the possibility of performing Raptor decoding at non-target sink devices is greater than that of target sink device. Therefore, non-target sink devices may consume more energy than the target sink device. However, it is very difficult to solve the fairness problem in the performance at sink devices because their battery status or power supply status can be different each other. Although the fairness problem in the performance of sink devices is very important, we will handle this problem in the future work because it is beyond the scope of this paper.

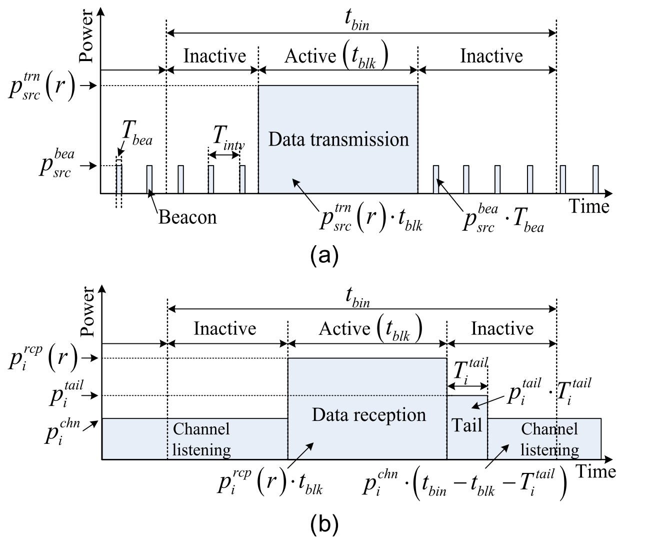

Fig. 3. Energy consumption patterns of WiFi network interfaces: (a) Source device and (b) sink device.

To solve the above problem with low computational complexity for real-time processing, we utilize the model-based estimation method.

### 3.2 Energy Consumption Model and Delay Model

In this section, we propose an energy consumption model of the WiFi network interface and energy consumption models and delay models for Raptor encoding and decoding processes.

#### 3.2.1 Energy Consumption Model for WiFi Network Interface

The general energy consumption patterns of the WiFi network interfaces at the source device and sink devices are presented in Fig. 3. Because the source device behaves like an AP, it periodically broadcasts beacon messages to the sink devices during the inactive state. The sink devices constantly listen to the wireless channel to overhear unicast transmission. When there is data to be received from the source device, the target sink device requests the data from the source device. Now, the source device transmits data using unicast transmission to the target device, and the non-target devices immediately overhear these data. After completing the data transmission, the source device and target sink device enter into the inactive state if there is no additional data for tail time [38]. However, non-target devices immediately go into the inactive state as soon as the data transmission is finished. Since only the active state (data transmission and reception) and inactive state (beacon transmission, tail, and channel listening) are considered in this paper, the energy consumption of the WiFi network interfaces during $t_{bin}$ is the sum of the shaded areas in Fig. 3. For $n_{blk}$ blocks encoded with the given $s$, $k$, and $c$, the energy consumption of the WiFi network interface at the source device and the energy consumption of the WiFi network interface at the $i$th sink device can be modeled as follows.

$$\tilde{e}_{src}(n_{blk},s,k) = p_{src}^{trn}(r)\cdot t_{blk} + p_{src}^{bea}\cdot(t_{bin}-t_{blk})\cdot(T_{bea}/T_{intv}), \tag{15}$$

$$\tilde{e}_i^{sink}(n_{blk},s,k) = p_i^{rcp}(r)\cdot t_{blk} + p_i^{tail}\cdot T_i^{tail} + p_i^{chn}\cdot\left(t_{bin}-t_{blk}-T_i^{tail}\right), \tag{16}$$

where $r$ is the data transmission rate, $p_{src}^{trn}(r)$ is the data transmission power of the source device, $t_{blk}$ is the transmission delay of Raptor encoding blocks (i.e., $(n_{blk}\cdot s\cdot k)/(c\cdot r)$), $p_{src}^{bea}$ is the beacon transmission power, $T_{bea}$ is the beacon transmission duration, $T_{intv}$ is the interval of the beacon transmission, $p_i^{rcp}(r)$ is the data reception power of the $i$th sink device, $p_i^{tail}$ is the tail power of the $i$th sink device, $p_i^{chn}$ is the channel listening power of the $i$th sink device, and $T_i^{tail}$ is the tail time of the $i$th sink device. $p_{src}^{trn}(r)$ and $p_i^{rcp}(r)$ are calculated using a simple linear model [38] as follows.

$$p_{src}^{trn}(r) = \beta_{src}\cdot r + \gamma_{src}, \tag{17}$$

$$p_i^{rcp}(r) = \beta_i^{sink}\cdot r + \gamma_i^{sink}, \tag{18}$$

where $\beta_{src}$ and $\gamma_{src}$ are the power model parameters of the source device, and $\beta_i^{sink}$ and $\gamma_i^{sink}$ are the power model parameters of the $i$th sink device ($\beta_{src}$, $\gamma_{src}$, $\beta_i^{sink}$, $\gamma_i^{sink} > 0$). In general, the WiFi network interface stays in the tail state with steady power consumption for a period after the data transfer is completed in order to reduce signaling overhead and latency [38]. The tail time is set by the device manufacturer.

Finally, the total energy consumption of WiFi network interfaces during $t_{bin}$ at all devices can be calculated using $\tilde{e}_{src}(n_{blk},s,k)$ and $\tilde{e}_i^{sink}(n_{blk},s,k)$ as follows.

$$\tilde{e}_{net}(n_{blk},s,k) = \tilde{e}_{src}(n_{blk},s,k) + \sum_{i=1}^{N_{sink}}\tilde{e}_i^{sink}(n_{blk},s,k). \tag{19}$$

#### 3.2.2 Energy Consumption and Delay Model for Raptor Encoding/Decoding Processes

For Raptor encoding and decoding processes, the XOR operation is the most dominant process [36], [37]. Thus, the energy consumption for Raptor encoding and decoding processes can be predicted based on the amount of XORed bytes, which is calculated by multiplying the symbol size by the number of symbol-level XOR operations. In the proposed system, we implemented Raptor codes using a single instruction multiple data (SIMD) technology [39] to improve the performance of the Raptor encoding and decoding processes. The SIMD is a well-known parallel processing technology that enables the parallel processing of multiple data with a single instruction, e.g., matrix summation and multiplication. Since the performance of the Raptor codes in the proposed system is affected by the SIMD-based implementation, we derive the SIMD-based energy consumption and delay models. The SIMD-based Raptor encoding energy consumption for a Raptor encoding block $\tilde{e}_{enc}(s,k,c)$ and Raptor decoding energy consumption for a Raptor encoding block at the $i$th sink device $\tilde{e}_i^{dec}(s,k)$ can be represented by

$$\tilde{e}_{enc}(s,k,c) = \rho_{enc}(s)\cdot\left\{s\cdot n_{xor}^{enc}(k,c)\right\}^{\mu_{enc}(s)}, \tag{20}$$

$$\tilde{e}_i^{dec}(s,k) = \rho_i^{dec}(s)\cdot\left\{s\cdot n_{xor}^{dec}(k)\right\}^{\mu_i^{dec}(s)}, \tag{21}$$

where $n_{xor}^{enc}(k,c)$ is the total number of symbol-level XOR operations required for the Raptor encoding, $n_{xor}^{dec}(k)$ is the total number of symbol-level XOR operations required for the Raptor decoding, $\rho_{enc}(s)$ and $\mu_{enc}(s)$ are the energy coefficients for the Raptor encoding at the source device, and $\rho_i^{dec}(s)$ and $\mu_i^{dec}(s)$ are the energy coefficients for the Raptor decoding at the $i$th sink device. The energy coefficients for the Raptor encoding and decoding processes are related to the computing power of the mobile device. These coefficients can be obtained by using the curve fitting method from the measured energy consumption of the Raptor encoding and decoding processes. In this paper, we adopt the Levenberg-Marquardt algorithm [46], which is a well-known curve fitting method for non-linear functions. Consequently, for $n_{blk}$ Raptor encoding blocks, the total energy consumption of Raptor encoding and decoding processes $\tilde{e}_{rap}(n_{blk}, s, k)$ can be calculated using $\tilde{e}_{enc}(s, k, c)$ and $\tilde{e}_i^{dec}(s, k)$ as follows.

$$\tilde{e}_{rap}(n_{blk}, s, k) = n_{blk} \cdot \tilde{e}_{enc}(s, k, c) + \sum_{i=1}^{N_{sink}} E_i(v_i, n_{blk}) \cdot \tilde{e}_i^{dec}(s, k), \tag{22}$$

$$E_i(v_i, n_{blk}) = \left\lceil n_{blk} \cdot \left\{ v_i \cdot plr_i^{uni} + (1 - v_i) \cdot plr_i^{over} \right\} \right\rceil. \tag{23}$$

Similarly, the SIMD-based Raptor encoding delay $\tilde{t}_{enc}(s, k, c)$ and Raptor decoding delay $\tilde{t}_i^{dec}(s, k)$ can be estimated based on the amount of XORed bytes, that is,

$$\tilde{t}_{enc}(s, k, c) = \sigma_{enc}(s) \cdot s \cdot n_{xor}^{enc}(k, c), \tag{24}$$

$$\tilde{t}_i^{dec}(s, k) = \sigma_i^{dec}(s) \cdot s \cdot n_{xor}^{dec}(k), \tag{25}$$

where $\sigma_{enc}(s)$ denotes the delay coefficients for Raptor encoding, and $\sigma_i^{dec}(s)$ denotes the delay coefficients for Raptor decoding at the $i$th sink device. The delay coefficients for the Raptor encoding and decoding processes are obtained by using the least square solution [47]. In the proposed system, all coefficients of energy models and delay models are measured and embedded at each source and sink device before connection is set up. Coefficients of sink device, such as $\rho_i^{dec}(s)$, $\mu_i^{dec}(s)$, and $\sigma_i^{dec}(s)$, are provided to the source device during the connection setup process.

In fact, the energy consumption of the proposed system and the coefficients of the energy models are strongly related to the hardware specification of the mobile device. Hence, it is very difficult to find the typical coefficients of the energy models. But several device manufacturers provide the power profile to estimate the device energy consumption [49]. If the power profile is offered by device manufacturer, then we can approximately calculate the coefficients of the energy models.

## 3.3 Parameter Determining Algorithm

In this section, we present the parameter determining algorithm to obtain a feasible solution. First, the target sink device and code rate determining algorithm is studied. Then, the Raptor encoding parameter selection algorithm is described in detail.

### 3.3.1 Target Sink Device and Code Rate Determining Algorithm

We provide the determining algorithm for $\vec{v}$ and $c$. In the target sink device and code rate determining problem, when the target sink device is fixed, the solution candidates of $c$ can be obtained by calculating the code rates which can achieve a successful Raptor decoding at a certain sink device. Thus, the optimal solution of $\vec{v}$ and $c$ that minimizes the given cost function $n_{pkt}^{cor}(\vec{v}, c)$ can be easily obtained by conducting a full search among all possible candidates of $c$ for all sink devices. Details of the target sink device and code rate determining algorithm are presented below.

*Step 0)* Empty the candidate parameter set $P_{cnd}^{tc}$.

*Step 1)* Set the candidate target sink device selection vector $\vec{v}_{cnd}$ by selecting the $j$th sink device as a target sink device $(1 \leq j, m \leq N_{sink})$.

$$\vec{v}_{cnd} = \left( v_1^{cnd}, v_2^{cnd}, \ldots, v_{N_{sink}}^{cnd} \right), \ v_m^{cnd} = \begin{cases} 1 & \text{if } j = m, \\ 0 & \text{otherwise.} \end{cases}$$

*Step 2)* Calculate code rate candidate $c_{cnd}$ for the $m$th sink device $(1 \leq m \leq N_{sink})$.

$$\begin{aligned} c_{cnd} &= \frac{k}{k_{pkt} \cdot n_m^{trs}}, \ k_{pkt} = \left\lfloor \frac{S_{pkt}}{s} \right\rfloor, \\ n_m^{trs} &= \left\lceil \frac{n_{pkt}^{\min}}{1 - \left\{ v_m^{cnd} \cdot plr_m^{uni} + \left(1 - v_m^{cnd}\right) \cdot plr_m^{over} \right\}} \right\rceil, \end{aligned} \tag{26}$$

where $k_{pkt}$ is the number of symbols in a packet, and $n_m^{trs}$ is the minimum number of transmitted packets required for successful Raptor decoding at the $m$th sink device.

*Step 3)* Calculate cost function $n_{pkt}^{cor}(\vec{v}_{cnd}, c_{cnd})$, and examine the constraint in Eq. (13). $\phi_i^{dec}(s, k, c)$ can be calculated by

$$\begin{aligned} \phi_i^{dec}(s, k, c) &= P\Big( X_i < n_{pkt}^{\min} \big| X \sim B\big( n_i^{recv}(v_i^{cnd}, c),\ 1 - plr_i \big) \Big) \\ &= \sum_{i=0}^{n_{pkt}^{\min}-1} \left( \binom{n_i^{recv}(v_i^{cnd}, c)}{i} (1 - plr_i)^i (plr_i)^{n_i^{recv}(v_i^{cnd}, c) - i} \right), \end{aligned} \tag{27}$$

$$plr_i = v_i^{cnd} \cdot plr_i^{uni} + \left(1 - v_i^{cnd}\right) \cdot plr_i^{over}, \tag{28}$$

where $X_i$ is the random variable of the number of received packets at the $i$th sink device, and $plr_i$ is the PLR at the $i$th sink device [36], [37]. If the current cost is smaller than the cost computed with the parameters in $P_{cnd}^{tc}$, then the candidate parameters in $P_{cnd}^{tc}$ are replaced with the current $(\vec{v}_{cnd}, c_{cnd})$.

*Step 4)* Repeat Step 2 and 3 until all possible $m$ are examined.

*Step 5)* If all possible $j$ are examined, then terminate the process with a solution in $P_{cnd}^{tc}$. Otherwise, go back to Step 1.

### 3.3.2 Raptor Encoding Parameter Selection Algorithm

We describe a method to determine the Raptor encoding parameters (i.e., $n_{blk}$, $s$, and $k$). In the Raptor encoding parameter selection problem, the available set of $s$ and $k$ are finite [37]. Moreover, $n_{blk}$ can be determined by calculating the maximum number of Raptor encoding blocks when $s$ and $k$ are given. Therefore, the solution that minimizes the given cost function $e_{pkt}^{sel}(n_{blk}, s, k)$ can be obtained by conducting a full search for all available set of $s$ and $k$. In fact, the solution obtained by the proposed algorithm is a near optimal solution for the optimal problem formulation, but it is a feasible solution with a low computational complexity for real-time processing. Details of the optimization procedures are presented below, and the flow chart of the overall parameter determining algorithm is shown in Fig. 4.

*Step 0)* Empty the candidate parameter set $P_{cnd}^{rep}$ and generate the combinations of $(s, k)$.

*Step 1)* Select one of the generated combinations of $(s, k)$.

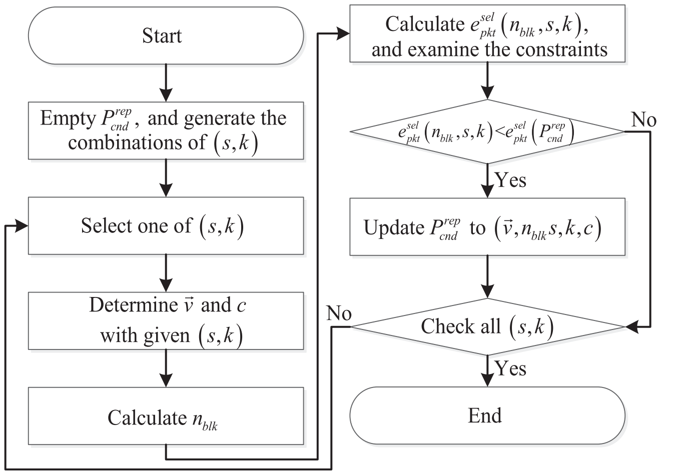
Start
Empty $P_{cnd}^{rep}$, and generate the combinations of $(s,k)$
Select one of $(s,k)$
Determine $\vec{v}$ and $c$ with given $(s,k)$
Calculate $n_{blk}$
Calculate $e_{pkt}^{sel}(n_{blk},s,k)$, and examine the constraints
$e_{pkt}^{sel}(n_{blk},s,k)<e_{pkt}^{sel}(P_{cnd}^{rep})$
No
Yes
Update $P_{cnd}^{rep}$ to $(\vec{v},n_{blk}s,k,c)$
Check all $(s,k)$
No
Yes
End

Fig. 4. Overall procedure of the parameter determining algorithm.

*Step 2)* Determine $\vec{v}$ and $c$ using the target sink device and code rate determining algorithm with the selected $(s,k)$.

*Step 3)* Calculate $n_{blk}$ as follows.

$$n_{blk} = \begin{cases} \frac{\tilde{t}_{buf} - t_{ott} - \max_{1\le i\le N_{sink}} \tilde{t}_i^{dec}(s,k)}{\tilde{t}_{enc}(s,k,c) + t_{blk}^{one}} & \text{if } t_{blk}^{one} \ge \max\limits_{1\le i\le N_{sink}} \tilde{t}_i^{dec}(s,k) \\ \frac{\tilde{t}_{buf} - t_{ott} - t_{blk}^{one}}{\tilde{t}_{enc}(s,k,c) + \max_{1\le i\le N_{sink}} \tilde{t}_i^{dec}(s,k)} & \text{otherwise,} \end{cases} \tag{29}$$

$$t_{blk}^{one} = \frac{s \cdot k}{c \cdot r}, \tag{30}$$

where $t_{blk}^{one}$ denotes the transmission delay for a Raptor encoding block, and $\tilde{t}_{buf}$ denotes the estimated buffered video playback time at the sink device. $\tilde{t}_{buf}$ is calculate by

$$\tilde{t}_{buf} = \min_{1\le i\le N_{sink}} \left( t_i^{buf} + t_i^{ott} \right), \tag{31}$$

where $t_i^{buf}$ denotes the measured buffered video playback time at the $i$-th sink device, and $t_i^{ott}$ is the one-way trip time between the source device and the $i$th sink device.

*Step 4)* Calculate the cost function $e_{pkt}^{sel}(n_{blk}, s, k)$, and examine the constraint in Eq. (14). If the current cost is smaller than the cost computed with the parameters in $P_{cnd}^{rep}$, and the constraint is satisfied, then the candidate parameters in $P_{cnd}^{rep}$ are replaced with the current parameters $(\vec{v}, n_{blk}, s, k, c)$.

*Step 5)* If all possible combinations of $(s,k)$ are examined, then terminate the process with the optimal solution $P_{cnd}^{rep}$. Otherwise, go back to Step 2.

## 4 Experimental Results

During the experiment, the proposed system is implemented using GStreamer [40] on a Linux-based single board computer, called ODROID [41], as shown in Fig. 5. The proposed system is examined over a real WiFi network with mobility. As illustrated in Fig. 6, we distributed one source device and four sink devices in our laboratory, and the source device moves around the laboratory during the experiment. For the real-time processing, two sink devices (Sink device #1 and #2) are running on ODROID-U3 equipped with a Samsung Exynos 4412 Prime Cortex-A9 Quad Core 1.7 GHz processor and 2GB RAM. Other two sink devices (Sink device #3 and #4) and a source device are running on ODROID-XU4 equipped with a Samsung Exynos 5422 Cortex-A15 Quad Core 2.0 GHz and Cortex-A7 Quad Core 1.4 GHz processor with 2 GB RAM. In the proposed system, Raptor codes are implemented based on [25] with SIMD technology, and the traffic shaping mechanism is implemented with reference to Linux Traffic Control [48].

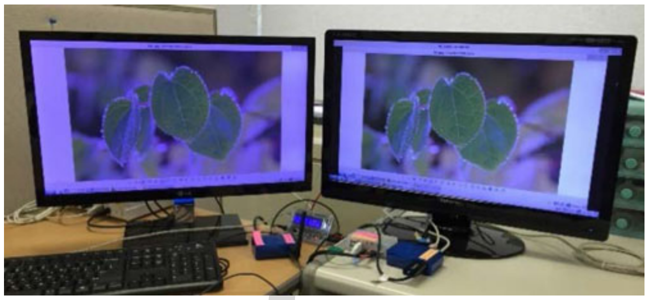

Fig. 5. Proposed system implemented on ODROID.

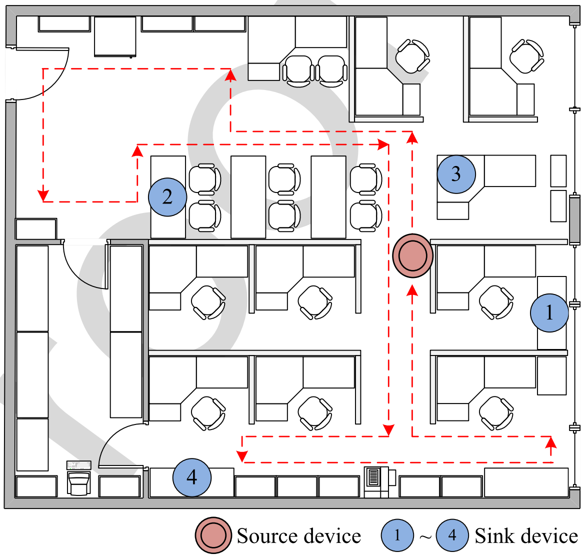

Fig. 6. Test environment of the proposed system (red dotted arrows indicate the moving route of the source device).

The experimental environment is set up as follows. For test mirroring contents, we use a stored video, gallery application that changes HD images every 5 seconds, and YouTube music video [42]. The stored video is encoded by H.264 with an average of 5 Mbps bit rate and 25 frames per second, and made by combining the Pedestrian Area, Rush Hour, and Sunflower videos [43]. The encoding structure is IPPPPPPPPPPP (i.e., 1 GOP consists of 12 frames). The packet size $S_{pkt}$ is set to 1,024 bytes. The set of symbol sizes and set of number of symbols are set to {64, 128, 256, 512} and {128, 192, 256, 320, 384, 448, 512}, respectively. To measure the consumed energy, an ODROID Smart Power, which is a power measurement tool for ODROID devices, is used [41]. Based on the power model measurement methods in [37], the energy consumption model parameters of the WiFi network interfaces are empirically measured by the ODROID Smart Power as presented in Table 2. Figs. 7 and 8 show the measured data and models obtained by using the curve fitting method. As shown in the figures, their differences are very small. The model coefficients obtained by using the curve fitting methods are

TABLE 2
Power Profiles of WiFi Network Interfaces

| Source device | | Sink device | |
|---|---|---|---|
| Parameter | Value | Parameter | Value |
| $\beta_{src}, \gamma_{src}$ | $2.90, 396.97$ | $\beta_i^{sink}, \gamma_i^{sink}$ | $2.67, 761.02$ |
| $p_{src}^{bea}$ | 370 mW | $p_i^{tail}$ | 720.56 mW |
| $T_{bea}$ | 100 ms | $p_i^{chn}$ | 420.54 mW |
| $T_{intv}$ | 100 ms | $T_i^{tail}$ | 150 ms |

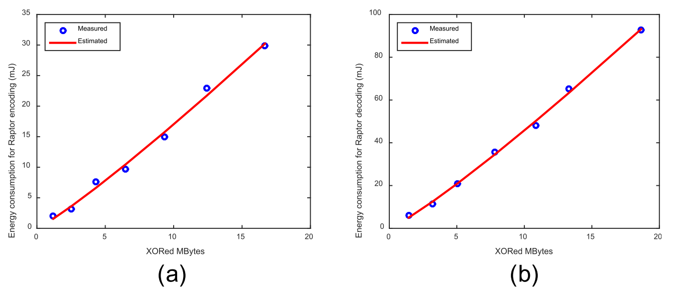

Fig. 7. Examples of the curve fitting for the energy consumption model of Raptor encoding/decoding when s = 128: (a) Energy consumption of Raptor encoding at source device. (b) Energy consumption of Raptor decoding at sink device #1.

TABLE 3
Coefficients of Energy Consumption and Delay Model for Raptor Encoding and Decoding

| $s$ | Encoding (Source device) | | | Decoding (Sink device #1) | | |
|---|---|---|---|---|---|---|
| | $\rho_{enc}(s)$ | $\mu_{enc}(s)$ | $\sigma_{enc}(s)$ | $\rho_i^{dec}(s)$ | $\mu_i^{dec}(s)$ | $\sigma_i^{dec}(s)$ |
| 64 | 0.00149 | 1.1560 | 0.6235 | 0.00447 | 1.3070 | 2.7300 |
| 128 | 0.00128 | 1.1230 | 0.4555 | 0.00331 | 1.1410 | 1.4486 |
| 256 | 0.00136 | 1.0210 | 0.3885 | 0.00170 | 1.1330 | 0.7950 |
| 512 | 0.00170 | 0.9302 | 0.3269 | 0.00157 | 1.0160 | 0.4855 |

presented in Table 3. These parameters are embedded in the implemented system.

For the performance comparison, the peak signal-to-noise ratio (PSNR) between the original screen content and the received screen content is adopted as an objective spatial video quality metric. The average PSNR denotes the average of PSNRs observed at four sink devices. Furthermore, the control packet overhead for feedback information is adopted as a system overhead metric. The control overhead $PO_{ctrl}$ is defined as follows:

$$PO_{ctrl} = \frac{n_{byte}^{ctrl}}{n_{pkt}^{data} \cdot S_{pkt} + n_{byte}^{ctrl}}, \qquad (32)$$

where $n_{pkt}^{data}$ is the number of transmitted data packets including video data packets, redundant packets, and retransmitted packet, and $n_{byte}^{ctrl}$ is the total size of the transmitted control packets from the source device. The control packets include all packets except for data packets.

## 4.1 Energy Consumption and Delay Model Verification

In this section, we present the model verification of the energy consumption and delay models. During the experiment, stored video is used as the test mirroring content. The experimental results are captured at the source device and sink device #1. First, we examine how well the energy

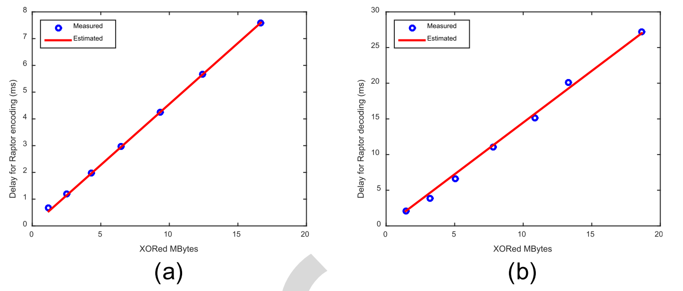

Fig. 8. Examples of the curve fitting for the delay model of Raptor encoding/decoding when s = 128: (a) Delay of Raptor encoding at source device. (b) Delay of Raptor decoding at sink device #1.

TABLE 4
Measured and Estimated Energy Consumption According to the Amount of Transmitted Data in a Bin

| Amount of transmitted data (Mbits) | Source device | | Sink device #1 | |
|---|---|---|---|---|
| | Measured energy (mJ) | Estimated energy (mJ) | Measured energy (mJ) | Estimated energy (mJ) |
| 5 | 3130.50 | 3011.96 | 5979.45 | 5491.83 |
| 10 | 3580.77 | 3502.33 | 6725.18 | 6947.69 |
| 15 | 3938.93 | 3844.42 | 7324.13 | 7219.24 |
| 20 | 4276.66 | 4431.41 | 8144.21 | 7609.94 |
| 25 | 4799.70 | 4879.67 | 8664.43 | 8380.34 |
| 30 | 5342.65 | 5018.58 | 9150.55 | 9377.65 |

consumption model of the WiFi network interface fits the observed experimental data. During the experiment, the bin duration and the transmission rate are set to 10 seconds and 5 Mbps, respectively. Table 4 presents the comparison between the measured and estimated energy consumption in terms of the amount of transmitted data in a bin. It is apparently observed that the estimated energy fits well with the measured energy. However, an estimation error still exists between the measured energy and estimated energy, due to inaccurate estimation of network conditions and processing power consumption. The average estimation error rates (i.e., $|estimated\ value - measured\ value| \times 100/estimated\ value$) at the source device and sink device are approximately 3.29 and 4.20 percent, respectively.

Now, we investigate the accuracy of the SIMD-based energy consumption and delay models for Raptor encoding and decoding processes. Table 5 shows the measured and estimated energy consumption of Raptor encoding and decoding processes according to the symbol size and the number of source symbols. The amount of XORed bytes is obtained by multiplying the symbol size by the total number of symbol-level XOR operations. It is obviously shown that the energy consumption of Raptor encoding and decoding processes increases as the amount of XORed bytes increases. The average estimation error rates of the energy consumption of the Raptor encoding and decoding processes are approximately 9.07 and 8.12 percent, respectively. Table 6 presents the measured and estimated Raptor encoding and decoding delays. The Raptor encoding and decoding delay linearly increase as the amount of XORed bytes increases. The average estimation error rates of the Raptor encoding and decoding delay are approximately 6.39 and 5.54 percent, respectively. Consequently, we can say that the Raptor codes energy consumption model and delay models have a good fit with the observed data.

TABLE 5
Measured and Estimated Raptor Encoding/Decoding Energy According to Symbol Parameters

| $s$ | $k$ | Encoding (Source device) | | Decoding (Sink device #1) | |
|---|---|---|---|---|---|
| | | Measured energy (mJ) | Estimated energy (mJ) | Measured energy (mJ) | Estimated energy (mJ) |
| 128 | 128 | 2.02 | 1.57 | 6.17 | 4.97 |
| | 192 | 3.14 | 3.64 | 11.41 | 12.56 |
| | 256 | 7.60 | 6.62 | 20.73 | 21.06 |
| | 320 | 9.69 | 10.48 | 35.76 | 34.45 |
| | 384 | 14.98 | 15.72 | 48.07 | 50.13 |
| | 448 | 22.89 | 21.73 | 65.09 | 63.42 |
| | 512 | 29.88 | 30.23 | 92.80 | 93.15 |
| 256 | 128 | 3.05 | 3.31 | 6.60 | 5.58 |
| | 192 | 6.25 | 7.12 | 14.45 | 13.99 |
| | 256 | 14.41 | 12.26 | 22.47 | 23.37 |
| | 320 | 19.07 | 18.60 | 37.13 | 38.10 |
| | 384 | 24.72 | 26.89 | 54.06 | 55.29 |
| | 448 | 36.47 | 36.08 | 72.81 | 69.82 |
| | 512 | 49.03 | 48.70 | 101.34 | 102.27 |
| 512 | 128 | 9.13 | 7.31 | 15.57 | 9.23 |
| | 192 | 14.42 | 14.70 | 20.68 | 21.06 |
| | 256 | 29.07 | 24.13 | 33.93 | 33.37 |
| | 320 | 33.98 | 35.27 | 53.98 | 51.73 |
| | 384 | 44.39 | 49.35 | 64.89 | 72.26 |
| | 448 | 64.23 | 64.52 | 89.94 | 89.09 |
| | 512 | 86.94 | 84.80 | 127.64 | 125.48 |

TABLE 6
Measured and Estimated Raptor Encoding/Decoding Delay According to Symbol Parameters

| $s$ | $k$ | Encoding (Source device) | | Decoding (Sink device #1) | |
|---|---|---|---|---|---|
| | | Measured delay (ms) | Estimated delay (ms) | Measured delay (ms) | Estimated delay (ms) |
| 128 | 128 | 0.67 | 0.55 | 2.13 | 2.07 |
| | 192 | 1.19 | 1.15 | 3.82 | 4.67 |
| | 256 | 1.97 | 1.97 | 6.63 | 7.34 |
| | 320 | 2.97 | 2.96 | 11.08 | 11.30 |
| | 384 | 4.24 | 4.25 | 15.13 | 15.70 |
| | 448 | 5.67 | 5.66 | 20.07 | 19.29 |
| | 512 | 7.57 | 7.60 | 27.21 | 27.02 |
| 256 | 128 | 1.11 | 0.93 | 2.54 | 2.27 |
| | 192 | 2.20 | 1.97 | 4.55 | 5.12 |
| | 256 | 3.58 | 3.35 | 7.48 | 8.06 |
| | 320 | 5.28 | 5.05 | 11.97 | 12.40 |
| | 384 | 7.38 | 7.24 | 17.05 | 17.23 |
| | 448 | 9.65 | 9.66 | 22.32 | 21.17 |
| | 512 | 12.69 | 12.96 | 29.36 | 29.66 |
| 512 | 128 | 1.99 | 1.56 | 3.21 | 2.78 |
| | 192 | 3.82 | 3.31 | 5.78 | 6.25 |
| | 256 | 6.14 | 5.65 | 9.26 | 9.84 |
| | 320 | 8.93 | 8.49 | 15.18 | 15.15 |
| | 384 | 12.40 | 12.19 | 20.55 | 21.05 |
| | 448 | 16.23 | 16.25 | 26.58 | 25.86 |
| | 512 | 21.30 | 21.80 | 36.19 | 36.23 |

## 4.2 Performance Verification According to Various Parameters

The performance of the proposed system according to various parameters is provided. During the experiment, the source device and all sink devices remain stationary in the laboratory. $T_{play}^{\max}$ is set to 500, 1000, and 1500 ms for the performance comparison according to $T_{play}^{\max}$. The stored video is used as the test mirroring content. First, we describe the performance comparison conducted with the fixed Raptor encoding parameters in Table 7. It is clearly shown that the total energy consumption of the source device and four sink devices decreases as the block size (i.e., $n_{blk} \cdot s \cdot k$) increases. This is because the WiFi network interface can spend more time in the inactive state by traffic shaping. However, the playback delay increases proportionally to the block size because more time is required to transmit the block and perform Raptor encoding and decoding processes. Based on this phenomenon, the proposed system dynamically adjusts the number of Raptor encoding blocks, the symbol size, and the number of source symbols by considering the network conditions, as shown in Fig. 9 ($T_{play}^{\max}$ is 500 ms). As shown in the figure, the proposed system selects $s$ and $k$ from the given set of the symbol sizes and set of number of symbols. The number of Raptor encoding blocks is adaptively determined according to the estimated network conditions and buffered video playback time. Moreover, as

TABLE 7
Performance Comparison with Fixed Raptor Encoding Parameters

| Param. setting | $n_{blk}$ | $s$ | $k$ | $T_{play}^{\max}$ (ms) | Total energy (J) | Avg. PSNR (dB) | Playback delay (ms) |
|---|---|---|---|---|---|---|---|
| Fixed param. | 32 | 64 | 128 | - | 2387.56 | 43.50 | 332 |
| | 8 | 128 | 256 | - | 2407.01 | 43.50 | 375 |
| | 2 | 256 | 512 | - | 2657.46 | 43.50 | 384 |
| | 64 | 64 | 128 | - | 2328.97 | 43.50 | 631 |
| | 16 | 128 | 256 | - | 2345.26 | 43.50 | 738 |
| | 4 | 256 | 512 | - | 2589.82 | 43.50 | 754 |
| | 128 | 64 | 128 | - | 2309.35 | 43.50 | 1283 |
| | 32 | 128 | 256 | - | 2314.23 | 43.50 | 1481 |
| | 8 | 256 | 512 | - | 2495.90 | 43.50 | 1488 |
| Adaptive param. | - | - | - | 500 | 2282.13 | 43.50 | 415 |
| | - | - | - | 1000 | 2230.32 | 43.50 | 974 |
| | - | - | - | 1500 | 2192.43 | 43.50 | 1463 |

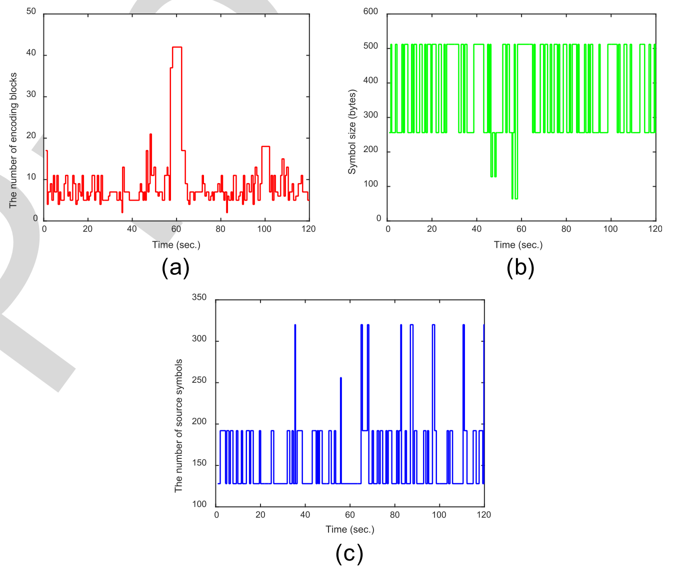

Fig. 9. Raptor encoding parameters of the proposed system: (a) Number of Raptor encoding blocks, (b) symbol size, and (c) number of source symbols (captured at source device).

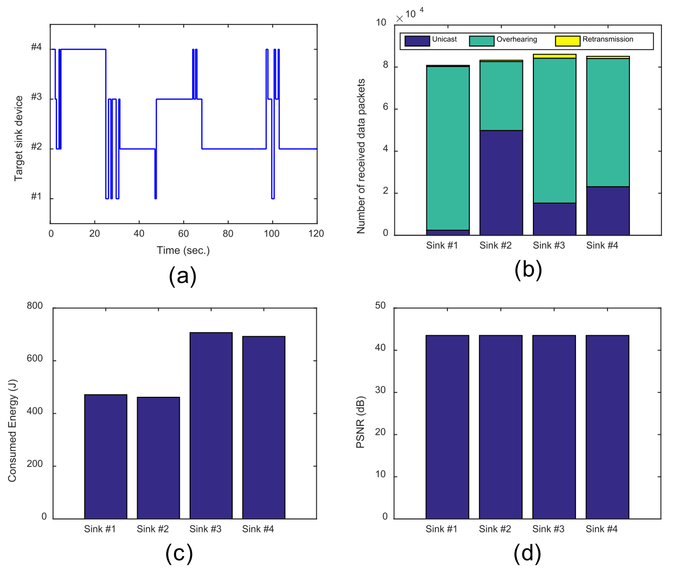

Fig. 10. Performance comparison according to sink device when the source device moves: (a) selected target sink device, (b) number of received data packets, (c) consumed energy, and (d) PSNR.

shown in Table 7, the playback delay between the source device and sink device can be controlled by $T_{play}^{max}$ (i.e., determined by user preference) in the proposed system. As $T_{play}^{max}$ increases, the playback delay increases while the energy consumption is reduced, and vice versa. Thus, the proposed system can achieve good energy efficiency while supporting a high-quality screen mirroring service.

We investigate the performance comparison of the proposed system according to the sink device when the source device is moving around. $T_{play}$ is set to 500ms, and the stored video is used as the test mirroring content. As shown in Fig. 10a, sink device #2 is selected as the target sink device more often than the others, so sink device #2 receives most of the data packets using unicast as shown in Fig. 10b. Meanwhile, sink device #1 receives most of the data packets using overhearing because the time selected as the target sink device is the shortest. However, since the PLR of overhearing at sink device #1 is very low, most of the data packets can be received without redundant packets and retransmission packets. On the other hand, sink device #3 requires more redundant packets and retransmission packets than sink device #1 and #2 in order to recover lost packets. Due to this, it consumes more energy than that of sink device #1 and #2, as shown in Fig. 10c. In particular, sink device #3 requires the largest amount of data and energy because its PLR is the lowest among the sink devices. All sink devices can successfully recover the lost packets and provide high quality screen mirroring service as shown in Fig. 10d.

### 4.3 Performance Comparison with Existing Systems

The performance of the proposed system is compared with that of three existing systems, namely Pseudo-broadcast [29], DirCast [12], and ACK-based multicast [44], which are modified slightly for our experiment. For a fair comparison, DirCast employs Raptor codes instead of the Reed Solomon codes [45], which is originally adopted in DirCast. During the experiment, the source device continuously moves around the laboratory, and all sink devices remain stationary, as shown in Fig. 6. For the experiment, we set $T_{play}^{max}$ to 500 ms based on [2]. First, we examine the cumulative curves of the received video data and consumed video data. During the experiment, the test mirroring contents are stored video. The results are presented in Fig. 11, and the arrows in the figure indicate the period of buffer underflow. As shown in the figure, it is obviously shown that the proposed system, Pseudo-broadcast, and DirCast can provide seamless video streaming without buffer underflow, whereas ACK-based multicast experience buffer underflows that result in frozen video.

The PSNR comparisons with existing protocols are presented in Fig. 12. It is clearly indicated in the figure that the

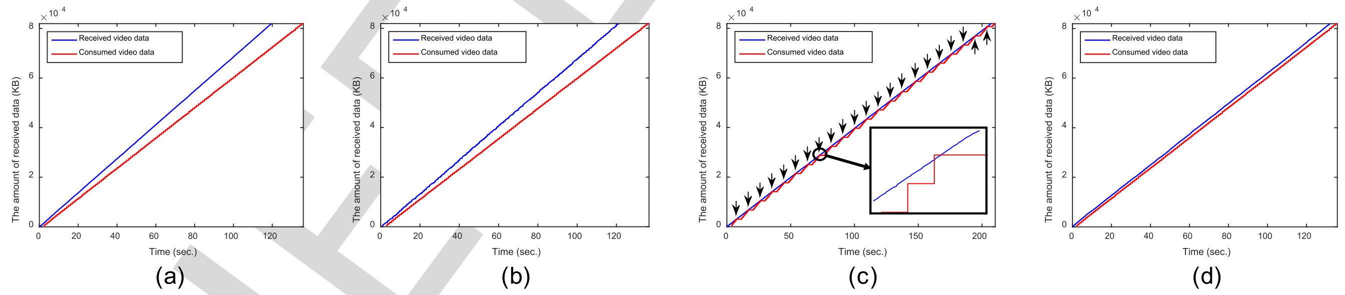

Fig. 11. Cumulative curves of received video data and consumed video data (captured at sink device #4): (a) Pseudo-broadcast, (b) DirCast, (c) ACK-based multicast, and (d) proposed system.

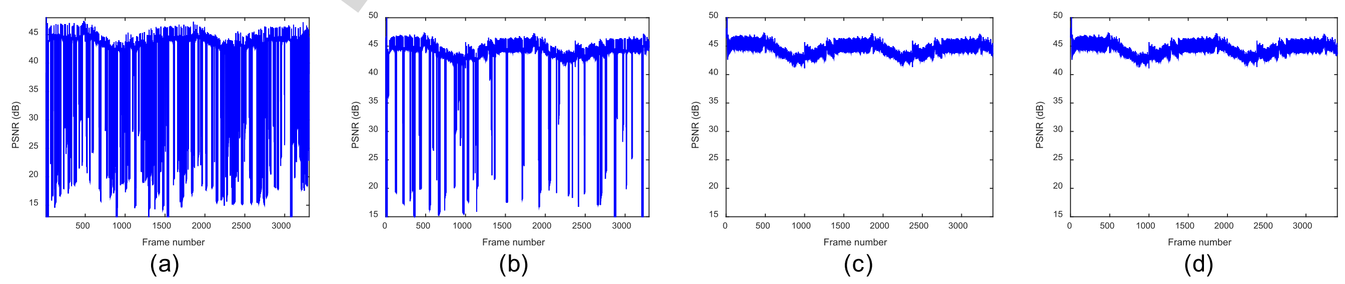

Fig. 12. PSNR comparison with existing systems (captured at sink device #4): (a) Pseudo-broadcast, (b) DirCast, (c) ACK-based multicast, and (d) proposed system.

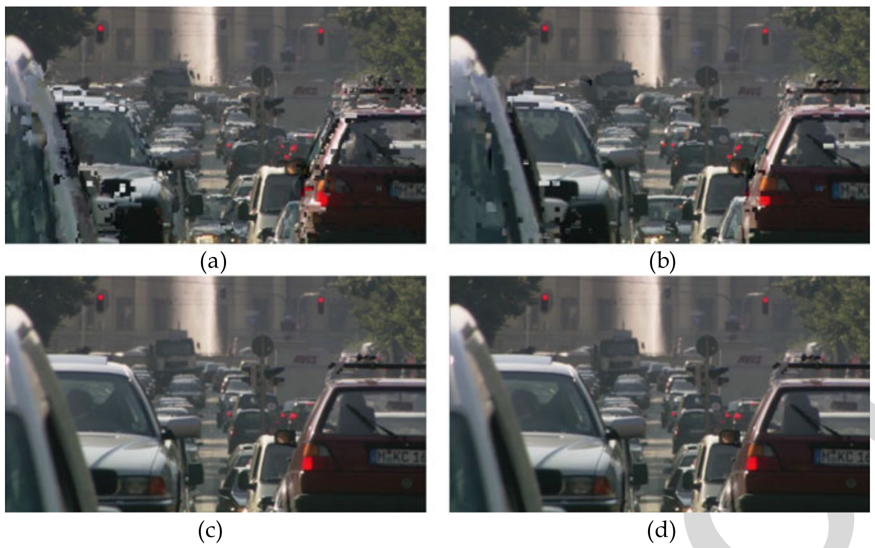

Fig. 13. Subjective video quality comparison of 1362nd frame (captured at sink device #4): (a) Pseudo-broadcast, (b) DirCast, (c) ACK-based multicast, and (d) proposed system.

TABLE 8
Energy and PSNR Results Per Device

| Mirroring Contents | System | Energy (J) | | | | | PSNR (dB) | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | Source | Sink #1 | Sink #2 | Sink #3 | Sink #4 | Sink #1 | Sink #2 | Sink #3 | Sink #4 |
| Stored video | Pseudo-broadcast | 568.37 | 359.59 | 349.12 | 539.39 | 523.68 | 21.43 | 35.85 | 35.85 | 30.74 |
| | DirCast | 709.37 | 468.47 | 462.46 | 702.70 | 693.68 | 41.81 | 40.70 | 41.32 | 42.39 |
| | ACK-based multicast | 1139.66 | 692.43 | 676.86 | 1038.64 | 1015.29 | 43.50 | 43.50 | 43.50 | 43.50 |
| | Proposed system | 706.45 | 471.20 | 461.51 | 706.80 | 692.26 | 43.50 | 43.50 | 43.50 | 43.50 |
| Gallery application | Pseudo-broadcast | 924.71 | 377.32 | 377.43 | 565.98 | 566.15 | 29.34 | 29.34 | 17.76 | 29.42 |
| | DirCast | 1289.88 | 513.10 | 527.34 | 769.66 | 791.01 | 33.83 | 41.16 | 41.27 | 36.13 |
| | ACK-based multicast | 1941.41 | 784.90 | 786.15 | 1177.34 | 1179.23 | 41.57 | 41.57 | 41.57 | 41.57 |
| | Proposed system | 1120.53 | 455.65 | 456.57 | 683.48 | 684.85 | 41.57 | 41.57 | 41.57 | 41.57 |
| YouTube music video | Pseudo-broadcast | 1032.99 | 424.07 | 436.14 | 636.11 | 654.20 | 32.56 | 29.90 | 29.90 | 19.30 |
| | DirCast | 1325.84 | 550.92 | 549.13 | 826.39 | 823.69 | 33.42 | 32.47 | 35.14 | 34.02 |
| | ACK-based multicast | 2073.18 | 811.87 | 802.00 | 1217.80 | 1203.00 | 41.44 | 41.44 | 41.44 | 41.44 |
| | Proposed system | 1160.93 | 478.89 | 481.51 | 718.34 | 722.27 | 41.44 | 41.44 | 41.44 | 41.44 |

TABLE 9
Summary of Performance Comparison with Existing Systems

| Mirroring Contents | System | Total Energy (J) | Avg. PSNR (dB) | Std. PSNR | Avg. Number of buffer underflows | Avg. Control overhead (%) |
|---|---|---|---|---|---|---|
| Stored video | Pseudo-broadcast | 2340.15 | 30.97 | 6.80 | 0 | 0.45 |
| | DirCast | 3036.68 | 41.56 | 0.72 | 0 | 0.55 |
| | ACK-based multicast | 4562.88 | 43.50 | 0 | 22 | 6.26 |
| | Proposed system | 3038.22 | 43.50 | 0 | 0 | 2.06 |
| Gallery application | Pseudo-broadcast | 2811.60 | 26.47 | 5.80 | 0 | 0.45 |
| | DirCast | 3891.00 | 38.10 | 3.72 | 0 | 0.53 |
| | ACK-based multicast | 5869.03 | 41.57 | 0 | 20 | 6.27 |
| | Proposed system | 3401.09 | 41.57 | 0 | 0 | 1.38 |
| YouTube music video | Pseudo-broadcast | 3183.50 | 27.92 | 5.88 | 0 | 0.53 |
| | DirCast | 4075.97 | 33.76 | 1.12 | 0 | 0.49 |
| | ACK-based multicast | 6107.85 | 41.44 | 0 | 21 | 6.25 |
| | Proposed system | 3561.94 | 41.44 | 0 | 0 | 0.97 |

video quality of Pseudo-broadcast is frequently and seriously degraded because Pseudo-broadcast does not support an error correction method. Although DirCast supports an error correction method, it is observed that the video quality is somewhat degraded because DirCast cannot finely adjust code rates considering the time-varying wireless network environment. Furthermore, DirCast cannot completely recover lost packets when unexpected extreme losses occur. Conversely, the proposed system and ACK-based multicast recover most lost packets successfully and support the screen mirroring service without any noticeable video quality degradation. For a subjective video quality comparison, the captured 1,362nd frame of the stored video are presented in Fig. 13. It is obviously shown that the video quality of both the proposed system and ACK-based multicast is much better than that of either Pseudo-broadcast or DirCast.

The energy and PSNR results per device and the summary of performance comparison with existing systems are shown in Tables 8 and 9, respectively. As shown in Tables 8 and 9, the proposed system and ACK-based multicast can support high-quality screen mirroring services with a 4.37 dB improvement on average compared to DirCast. The PSNR of the proposed system is 32.59 percent higher than Pseudo-broadcast and 10.45 percent higher than DirCast. Moreover, the proposed system and ACK-based multicast provide equal level of PSNR quality for all sink devices because they utilize lost packet recovery schemes. In the ACK-based multicast, the source device repeatedly transmits the lost video data until all sink devices transmit an ACK message that notifies the source device of successful reception. Thus, ACK-based multicast can provide reliable screen mirroring service without video quality degradation; however, it requires more time to receive the video data because the transmission of the next video data can be delayed at the source device until the source device receives ACK messages for the current video data from all sink devices. During the experiment, underflow occurs 21 times on average for ACK-based multicast. Meanwhile, in the case of the proposed system, most lost packets are recovered by the Raptor codes with no delay of retransmission requests and a small number of unexpected lost packets are recovered by the NACK-based retransmission request scheme with minimal delays. Thus, the proposed system cannot only provide reliable screen mirroring service, but can also provide seamless screen mirroring services without buffer underflow. Moreover, the proposed system requires a reasonable amount of control overhead compared to existing systems. As shown in Table 9, the control overhead of the proposed system is average four times lower than ACK-based multicast. As shown in Tables 8 and 9, the proposed system consumes less energy than the FEC-based systems, i.e., DirCast and ACK-based multicast. The proposed system can save 8.38 percent of energy consumption compared to DirCast. In addition, it is shown that the proposed system can provide an energy saving of 39.05 percent compared to ACK-based multicast while providing a similar level of video quality. Consequently, the proposed system can provide a reliable and energy-efficient hybrid screen mirroring service with relatively low control overhead.

## 5 Conclusion

In this paper, we have proposed a reliable and energy-efficient hybrid screen mirroring multicast system for sharing high-quality screen mirroring service among adjacent sink devices. In the proposed system, systematic Raptor codes and NACK-based retransmission are employed to reduce the video quality degradation over an error-prone WiFi network. The proposed system not only shapes the screen mirroring traffic, but also determines the target sink device and Raptor encoding parameters while considering the energy consumption of the source device and sink devices. The proposed system has been fully implemented in Linux-based single board computers, and tested over a real WiFi network. Experimental results show that the proposed system can provide energy savings of 39.05 percent compared to ACK-based multicast systems while providing the same level of video quality. Furthermore, the proposed system can provide high-quality screen mirroring without noticeable video quality degradation compared to existing systems.

## Acknowledgments

This research was supported by the Basic Science Research Program through the National Research Foundation of Korea(NRF) funded by the Ministry of Science, ICT & Future Planning(NRF-2016R1A2B4007812).

## References

[1] Screen mirroring awareness reaches 40 percent of smartphone and tablet owners. [Online]. Available: https://www.npd.com/wps/portal/npd/us/news/press-releases/screen-mirroring-awareness-reaches-40-percent-of-smartphone-and-tablet-owners-according-to-the-npd-group/, Accessed on: 2017.

[2] C. Hsu, T. Tsai, C. Huang, C. Hsu, and K. Chen, "Screencast dissected: Performance measurements and design considerations," in *Proc. ACM Conf. Multimedia Syst.*, Mar. 2015, pp. 177–188.

[3] C. F. Lai, H. Wang, H.-C. Chao, and G. Nan, "A network and device aware QoS approach for cloud-based mobile streaming," *IEEE Trans. Multimedia*, vol. 15, no. 4, pp. 747–757, Jun. 2013.

[4] S. Wang and S. Dey, "Adaptive mobile cloud computing to enable rich mobile multimedia applications," *IEEE Trans. Multimedia*, vol. 15, no. 4, pp. 870–883, Jun. 2013.

[5] C.-Y. Huang, K.-T. Chen, D.-Y. Chen, H.-J. Hsu, and C.-H. Hsu, "GamingAnywhere: The first open source cloud gaming system," *ACM Trans. Multimedia Comput.*, vol. 10, no. 1, pp. 173–185, Jan. 2014.

[6] Use AirPlay or AirPlay Mirroring on your iPhone, iPad, or iPod touch. [Online]. Available: https://support.apple.com/en-us/HT204289/, Accessed on: 2017.

[7] Chromecast. [Online]. Available: https://www.google.com/intl/en_us/chromecast/?utm_source = chromecast.com/, Accessed on: 2017.

[8] MirrorOp. [Online]. Available: http://www.mirrorop.com/, Accessed on: 2017.

[9] Splashtop. [Online]. Available: http://www.splashtop.com/, Accessed on: 2017.

[10] Wi-Fi Certified Miracast. [Online]. Available: https://www.wi-fi.org/discover-wi-fi/wi-fi-certified-miracast/, Accessed on: 2017.

[11] S. Sen, N. K. Madabhushi, and S. Banerjee, "Scalable WiFi media delivery through adaptive broadcasts," in *Proc. USENIX Symp. Netw. Syst. Des. Implementation*, Apr. 2010, pp. 191–204.

[12] R. Chandra, et al., "DirCast: A practical and efficient Wi-Fi multicast system," in *Proc. IEEE Int. Conf. Netw. Protocols*, Oct. 2009, pp. 161–170.

[13] J. C. MacKay, "Fountain codes," *IEEE Proc. Commun.*, vol. 152, no. 6, pp. 1062–1068, Dec. 2005.

[14] S. Ahmad, R. Hamzaoui, and M. M. Al-Akaidi, "Unequal error protection using fountain codes with applications to video communication," *IEEE Trans. Multimedia*, vol. 13, no. 1, pp. 92–101, Feb. 2011.

[15] Z. Luo, L. Song, S. Zheng, and N. Ling, "Raptor codes based unequal protection for compressed video according to packet priority," *IEEE Trans. Multimedia*, vol. 15, no. 8, pp. 2208–2213, Dec. 2013.

[16] C.-H. Lin, C.-K. Shieh, and W.-S. Hwang, "An access point-based FEC mechanism for video transmission over wireless LANs," *IEEE Trans. Multimedia*, vol. 15, no. 1, pp. 195–206, Jan. 2013.

[17] A. Hameed, R. Dai, and B. Balas, "A decision-tree-based perceptual video quality prediction model and its application in FEC for wireless multimedia communications," *IEEE Trans. Multimedia*, vol. 18, no. 4, pp. 764–774, Apr. 2016.

[18] Why Digital Fountain's Raptor Code Is Better Than Reed Solomon Erasure Codes For Streaming Applications, Digital Fountain, Inc., Fremont, CA, USA, 2005.

[19] M. A. Hoque, M. Siekkinen, and J. K. Nurminen, "TCP receive buffer aware wireless multimedia streaming: An energy efficient approach," in *Proc. ACM Workshop Netw. Operating Syst. Support Dig. Audio Video*, Feb. 2013, pp. 13–18.

[20] M. A. Hoque, M. Siekkinen, and J. K. Nurminen, "On the energy efficiency of proxy-based traffic shaping for mobile audio streaming," in *Proc. IEEE Consumer Commun. Netw. Conf.*, Jan. 2011, pp. 891–895.

[21] J. Korhonen and Y. Wang, "Power-efficient streaming for mobile terminals," in *Proc. ACM Workshop Netw. Operating Syst. Support Dig. Audio Video*, Jun. 2005, pp. 39–44.

[22] A. Shokrollahi, "Raptor codes," *IEEE Trans. Inf. Theory*, vol. 52, no. 6, pp. 2551–2567, Jun. 2006.

[23] C. Hsu, C. Fan, T. Tsai, C. Huang, C. Hsu, and K. Chen, "Toward adaptive screencast platform: measurement and optimization," *ACM Trans. Multimedia Comput. Commun. Appl.*, vol. 7, no. 3, Aug. 2015.

[24] S. Chandra, J. Boreczky, and L. Rowe, "High performance many-to-many Intranet screen sharing with DisplayCast," *ACM Trans. Multimedia Comput., Commun. Appl.*, vol. 10, no. 2, Feb. 2014.

[25] C. Zhang, X. Zhang, and R. Chandra, "Energy efficient wiFi display," in *Proc. ACM Conf. Mobile Syst. Appl. Services*, Nov. 2015, pp. 405–418.

[26] H. Ha, P. Bae, K. Lim, J. Ko, and Y. Ko, "Poster Abstract: Mobile contents on the big screen: Adaptive frame filtering for mobile device screen sharing," in *Proc. ACM Conf. Embedded Netw. Sensor Syst.*, Nov. 2014.

[27] P. Bae, J. Ko, J. Ryu, and Y. Ko, "Poster abstract: Screen dynamics analysis-based adaptive frame skipping for efficient mobile screen sharing," in *Proc. ACM Conf. Inf. Process. Sens. Netw.*, Apr. 2015.

[28] N. Choi, Y. Seok, T. Kwon, and Y. Choi, "Leader-based multicast service in IEEE 802.11v networks," in *Proc. IEEE Consumer Commun. Netw. Conf.*, Jan. 2010, pp. 1–5.

[29] Y. Park, C. Jo, S. Yun, and H. Kim, "Multi-room IPTV delivery through pseudo-broadcast over IEEE 802.11 links," in *Proc. IEEE Vehicular Technol. Conf. Spring*, May 2010, pp. 1–5.

[30] H.-T. Chiao, S.-Y. Chang, K.-M. Li, Y.-T. Kuo, and M.-C. Tseng, "WiFi multicast streaming using AL-FEC inside the trains of high-speed rails," in *Proc. IEEE Int. Symp. Broadband Multimedia Syst. Broadcasting*, Jun. 2012, pp. 1–6.

[31] M. Choi, W. Sun, J. Koo, and S. Choi, "Reliable video multicast over Wi-Fi networks with coordinated multiple APs," in *Proc. IEEE Conf. Comput. Commun.*, Apr. 2014, pp. 424–432.

[32] M. Siekkinen, M. A. Hoque, J. K. Nurminen, and M. Aalto, "Streaming over 3G and LTE: how to save smartphone energy in radio access network-friendly way," in *Proc. ACM Workshop Mobile Video*, Feb. 2013, pp. 13–18.

[33] H. Shen and Q. Qiu, "User-aware energy efficient streaming strategy for smartphone based video playback applications," in *Proc. Des. Autom. Test Europe Conf. Exhib.*, Mar. 2013, pp. 258–261.

[34] C. Poellabauer and K. Schwan, "Energy-aware traffic shaping for wireless real-time applications," in *Proc. IEEE Real-Time Embedded Technol. Appl. Symp.*, May 2004, pp. 48–55.

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

[36] O. C. Kwon, Y. Go, and H. Song, "An energy-efficient multimedia streaming transport protocol over heterogeneous wireless networks," *IEEE Trans. Vehicular Technol.*, vol. 65, no. 8, pp. 6518–6531, Aug. 2016.

[37] O. C. Kwon, Y. Go, Y. Park, and H. Song, "MPMTP: Multipath multimedia transport protocol using systematic raptor codes over wireless networks," *IEEE Trans. Mobile Comput.*, vol. 14, no. 9, pp. 1903–1916, Sep. 2015.

[38] 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 *Proc. ACM Conf. Mobile Syst. Appl. Services*, Jun. 2012, pp. 225–238.

[39] ARM NEON. [Online]. Available: http://www.arm.com/products/processors/technologies/neon.php/, Accessed on: 2017.

[40] GStreamer. [Online]. Available: https://gstreamer.freedesktop.org/, Accessed on: 2017.

[41] ODROID. [Online]. Available: http://www.hardkernel.com/main/main.php/, Accessed on: 2017.

[42] YouTube, A little girl MV. [Online]. Available: https://www.youtube.com/watch?v=bLoO0FSXncg/, Accessed on: 2017.

[43] Video test sequence. [Online]. Available: http://cs-nsl-wiki.cs.surrey.sfu.ca/wiki/Video_Library_and_Tools/, Accessed on: 2017.

[44] H. R. Oh, D. O. Wu, and H. Song, "An effective mesh-pull-based P2P video streaming system using fountain codes with variable symbol sizes," *Comput. Netw.*, vol. 55, no. 12, pp. 2746–2759, Aug. 2011.

[45] I. S. Reed and G. Solomon, "Polynomial codes over certain finite fields," *J. Soc. Ind. Appl. Math.*, vol. 8, no. 2, pp. 300–304, 1960.

[46] P. C. Hansen, V. Pereyra, and G. Scherer, *Least Squares Data Fitting with Applications*. Baltimore, MD, USA: Johns Hopkins Univ. Press, 2012.

[47] A. Howard, *Elementary Linear Algebra*. Hoboken, NJ, USA: Wiley, 2010.

[48] Linux Traffic Control. [Online]. Available: http://tldp.org/HOWTO/Traffic-Control-HOWTO/index.html, Accessed on: 2017.

[49] Power Profiles for Android. [Online]. Available: https://source.android.com/devices/tech/power/, Accessed on: 2017.

**Yunmin Go** received the BS degree from the School of Computer Science and Electrical Engineering, Handong Global University, Korea, in August 2010. Currently, he is working toward the PhD degree in the Division of IT Convergence and Engineering, Pohang University of Science and Technology (POSTECH), Korea. His research interests are in the areas of HTTP adaptive streaming, channel coding-based streaming, and energy-efficient wireless networks.

**Hwangjun Song** received the BS and MS degrees from the Department of Control and Instrumentation (EE), Seoul National University, Korea, in 1990 and 1992, respectively, and the PhD degree in electrical engineering-systems from the University of Southern California, Los Angeles, CA, USA, in 1999. From 1995 to 1999, he was a research assistant in SIPI (Signal and Image Processing Institute) and IMSC (Integrated Media Systems Center), University of Southern California. From 2000 to 2005, he was an assistant professor/vice dean of admission affairs at Hongik University, Seoul, Korea. Since Feb. 2005, he has been in the Department of Computer Science and Engineering, POSTECH (Pohang University of Science and Technology), Korea. He received the Haedong Paper Award from the Korean Institute of Communication Science in 2005. He was a courtesy associate professor with the University of Florida in 2011-2012. He served as a vice president of the Korean Institute of Information Scientists and Engineers, in 2016-2017. He is an APSIPA distinguished lecturer for 2017-2018. He is an editorial board member of the *Journal of Visual Communication and Image Representation* and an associate editor of the *Journal of Communications and Networks*, and served as an editorial board member of the *International Journal of Vehicular Technology* and a guest editor of the special issue on network technologies for emerging broadband multimedia services in the *Journal of Visual Communication and Image Representation* and the special issue on "wireless & mobile networks" in the *International Journal of Ad Hoc and Ubiquitous Computing*. His research interests include multimedia signal processing and communication, image/video compression, digital signal processing, network protocols necessary to implement functional image/video applications, and control system.