Dynamic Retry Adaptation Scheme to Improve Transmission of H.264 HD Video over 802.11 Peer-to-Peer Networks

Mohammed Sinky, Ben Lee, Tae-Wook Lee, Chang-Gone Kim, and Jong-Keun Shin

This paper presents a dynamic retry adaptation scheme for H.264 HD video, called DRAS.264, which dynamically adjusts the retry limits of frames at the medium access control (MAC) layer according to the impact those frames have on the streamed H.264 HD video. DRAS.264 is further improved with a bandwidth estimation technique, better prediction of packet delays, and expanded results covering multi-slice video. Our study is performed using the Open Evaluation Framework for Video Over Networks as a simulation environment for various congestion scenarios. Results show improvements in average peak signal-to-noise ratios of up to 4.45 dB for DRAS.264 in comparison to the default MAC layer operation. Furthermore, the ability of DRAS.264 to prioritize data of H.264 bitstreams reduces error propagation during video playback, leading to noticeable visual improvements.

Keywords: Wireless, network, H.264, streaming, multimedia.

Manuscript received Oct. 31, 2014; revised July 22, 2015; accepted July 30, 2015.

An earlier version of this work was published in the 8th International Conference on Ubiquitous Information Management and Communication (ICUIMC 2014).

This work was supported in part by LG Display Co., Korea and Ministry of Education Science and Technology (MEST) and the Korean Federation of Science and Technology Societies (KOFST).

Mohammed Sinky (corresponding author, mhsinky@uqu.edu.sa) is with the Computer Engineering Department, Umm Al-Qura University, Makkah, Saudi Arabia.

Ben Lee (benl@eecs.orst.edu) is with the School of Electrical Engineering and Computer Science, Oregon State University, Corvallis, USA.

Tae-Wook Lee (twlee@lgdisplay.com), Chang-Gone Kim (cgkim02@lgdisplay.com), and Jong-Keun Shin (rgbshin@lgdisplay.com) are with LG Display Co. Ltd., LCD Laboratory, Paju, Rep. of Korea.

I. Introduction

Peer-to-peer HD video streaming between smartphones, tablets, set-top boxes, and other mobile devices over WLANs has become an important enabling technology for home entertainment and N-screen applications. However, packet loss and delay are two major factors that affect the quality of video streams for these applications. The sensitivity of real-time video to these factors poses challenges for current enabling products, such as Intel WiDi [1] and Apple Airplay [2]. Although WLAN-based video streaming solutions have quickly emerged on the market, providing smooth playback of HD content is becoming more challenging as the popularity of peer-to-peer video streaming increases. Due to the lossy nature of wireless media, the IEEE 802.11 medium access control (MAC) layer provides built-in reliability by performing retransmissions of lost packets up to some fixed limit, referred to as the *retry limit* [3]. However, too many retransmissions can cause packet delays that lead to violation of video playback deadlines. Also, packets of compressed video carry varying levels of importance within the context of video reconstruction [4]. Therefore, more time may be spent delivering a lower priority packet at the expense of delaying other packets that have a higher visual impact on displayed video.

This problem was addressed in our prior work using an adaptive MAC layer retransmission scheme tailored to H.264 videos, known as *Dynamic Retry Adaption Scheme for H.264* (DRAS.264) [5]. In this scheme, H.264-compliant bitstreams are parsed in real time to evaluate packet priorities and then dynamically adjust their retry limits accordingly. Specifically, DRAS.264 assigns higher retry limits to any packet containing a slice header as well as packets that hold I-frame slice data. Slice headers impact visual quality the most, followed by the data present in I-frames. Therefore, prioritizing slice headers gives a decoder a better chance to reconstruct video frames, while protecting I-frame data prevents error propagation. All other packets of the bitstream are assigned lower retry limits to avoid unnecessary retransmissions of low-impact data.

This paper extends our prior work on DRAS.264 by applying DRAS.264 to multi-slice videos and incorporating bandwidth estimation. Therefore, in addition to providing a discussion of the basic DRAS.264 scheme, the specific contributions of the paper are as follows: 1) application of a new bandwidth estimation technique to dynamically enable or disable the DRAS.264 retry assignment (RA) process, 2) improvements to the RA mechanism of DRAS.264 by supplying the number of packets contained per slice to the MAC layer via cross-layer communication, and 3) performance evaluation of videos encoded using multi-slices, which is a built-in feature of the H.264 codec to allow for better error resiliency in networked environments.

This paper is organized as follows. Section II provides a background on the contention problem at the MAC layer and the important error-resilience features of the H.264 codec. Section III discusses the related work. The core of the original DRAS.264 algorithm augmented with the new bandwidth estimation process and improved RA method is presented in Section IV. Section V shows a comparison between the performance of the newly modified DRAS.264 and the default MAC layer operation. Finally, Section VI concludes the paper and discusses possible future work.

II. Background

Any shared medium requires serialized access to prevent contention. The distributed coordination function (DCF) of the 802.11 MAC protocol governs access to shared wireless media. Since stations (STAs) must be given atomic access, multiple STAs in a network cause additional delay during transmission. For real-time video, high end-to-end delays can lead to deadline violations and thus video quality degradation. This section covers the delays associated with the DCF. Furthermore, the implications of delayed and lost packets are discussed in terms of how they affect video reconstruction for H.264-encoded video.

1. DCF Operation

The IEEE 802.11 DCF relies on specific time intervals. The

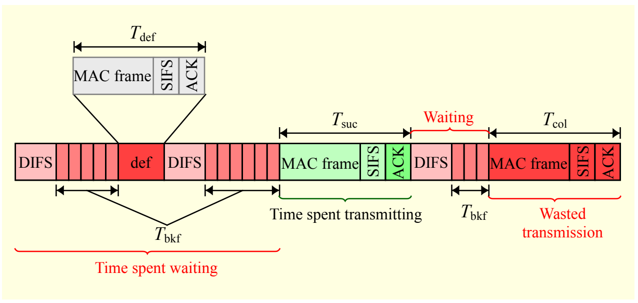


Fig. 1. IEEE 802.11 DCF Timing.

shortest of these intervals is referred to as a time slot (T_{slot}), and STAs check the availability of the medium in integrals of this time period [3]. The next shortest interval is the short interframe space (T_{SIFS}), which is a mandatory waiting period between the successful delivery of a MAC frame and when its corresponding acknowledgement (ACK) frame is sent in response. Finally, the DCF interframe space (T_{DIFS}), which is given as $T_{\mathrm{DIFS}} = T_{\mathrm{SIFS}} + 2\times T_{\mathrm{slot}}$, is the duration for which the medium must be sensed idle before a new MAC frame can begin transmission. DCF avoids collisions by assigning a random backoff (BO) variable representing the number of time slots that an STA must wait after T_{DIFS} before it can start transmission. The BO variable is selected from the range [0, CW], where CW is known as the contention window. The CW doubles for each failed transmission attempt until it reaches a standard-defined maximum value.

Figure 1 illustrates an example timing of DCF, which consists of the following four main states: success, collision, BO, and deferred. In Fig. 1, the STA spends a total of 14 time slots in the BO state, represented by T_{bkf} (11 during the first transmission and 3 during the second transmission). Additionally, it transitions to the deferred state (T_{def}) when another STA gains access to the medium. During transmission, either successful or unsuccessful delivery of the corresponding ACK frame is the difference between the success (T_{suc}) and collision (T_{col}) states. For the second transmission in Fig. 1, another STA has seized the medium at the same time causing collision, thus wasting the time spent transmitting.

To simplify the timing calculations for each state, all packets are assumed to be of equal length. Therefore, T_{suc}, T_{col}, and T_{def} are equivalent and are given as

$$T_{\mathrm{suc}} = T_{\mathrm{col}} = T_{\mathrm{def}} = T_{\mathrm{frm}} + T_{\mathrm{SIFS}} + T_{\mathrm{ACK}}\,, \tag{1}$$

where (T_{ACK}) is the time needed for an ACK frame to be delivered. Note that T_{bkf} for the rth retry, $T_{\mathrm{bkf}}(r)$, only depends on the random time slot chosen for that attempt, BO_r, Thus,

$$T_{\mathrm{bkf}}(r) = \mathrm{BO}_r \times T_{\mathrm{slot}} \in (\mathrm{CW}_r \times T_{\mathrm{slot}}). \tag{2}$$

A summary of the time durations for the defined intervals of the 802.11 DCF is provided in Table 1.

Table 1. Important time durations of 802.11 DCF scheme.

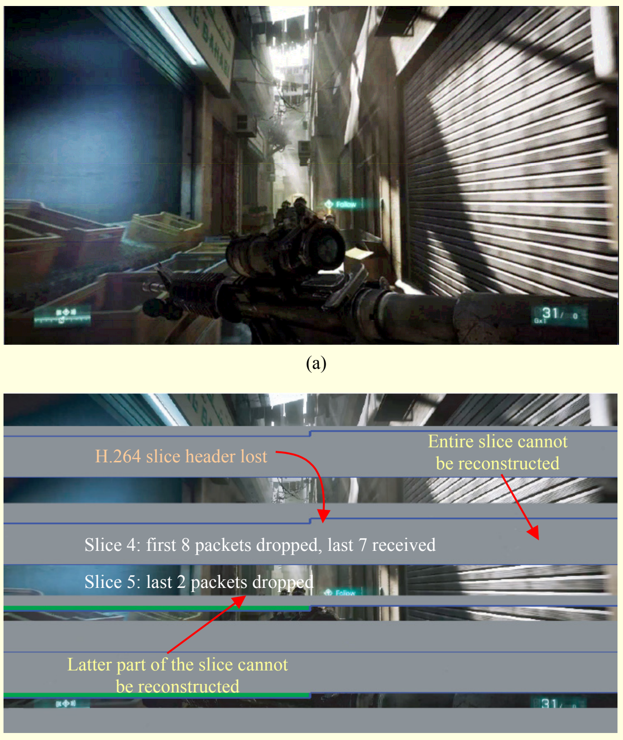
State	Values			
	802.11a	802.11b	802.11g	802.11n
T_{SIFS}	16 µs	10 µs	10 µs	10 µs
T_{DIFS}	34 µs	50 µs	28 µs	28 µs
T_{slot}	9 µs	20 µs	9 µs	9 µs
T_{ACK}	Data rate dependent			
T_{frm}	Payload size + data rate dependent			
SIFS: short interframe space, DIFS: DCF interframe space, ACK: MAC layer acknowledgement frame, frm: MAC layer data frame				

2. H.264 Codec

The H.264 codec, also referred to as Advanced Video Coding (AVC), is today's *de facto* standard for video compression [6]. Like all video codecs, H.264/AVC uses predictive methods to reconstruct video sequences; however, it exhibits a much higher coding efficiency than its predecessors. Encoded videos consist of sequences of groups of pictures (GOP), which are sets of coded frames placed in decoding order. Packet loss within a GOP sequence gives rise to error propagation, particularly when it happens early in the GOP sequence. This is because later frames in a GOP sequence reference earlier frames. The basic units operated on by the H.264 video codec are 16 × 16 pixel regions referred to as macroblocks (MBs). MBs are labeled according to whether bidirectional motion compensation (B-frames), previous frame motion compensation (P-frames), or intra prediction (I-frame) techniques are used. MBs may also be grouped into spatial regions of frames known as *slices*, which is a feature of the H.264 codec designed for error resiliency and parallel processing of images. Similar to the frame types, the three main slice types for H.264 encoded video are I-, P-, and B-slices. Additionally, a special type of intra prediction slice exists, which is known as an Instantaneous Decoder Refresh (IDR) slice. IDR-slices prevent decoders from referencing slices of earlier frames and always initiate new GOP sequences.

H.264 is composed of two layers — the video coding layer (VCL) and the network abstraction layer (NAL). The VCL is the compressed video bitstream. The NAL encapsulates the VCL with additional information, making it suitable for transmission over existing packet-based networks [7].

A slice-based representation of an H.264 video frame is given in Fig. 2, which shows an original video frame and the corresponding received frame with packet loss for an 8-slice video. Each slice maps to a spatial region within the frame whose boundaries are indicated by the solid blue lines in



(a)
H.264 slice header lost
Entire slice cannot be reconstructed
Slice 4: first 8 packets dropped, last 7 received
Slice 5: last 2 packets dropped
Latter part of the slice cannot be reconstructed
(b)

Fig. 2. Effect of packet loss on H.264 video reconstruction: (a) original frame and (b) received frame.

Fig. 2(b). The solid grey blocks correspond to missing information as a result of packet loss. A decoder will attempt to hide these areas using error concealment (EC) techniques [8]; however, the frame in Fig. 2(b) is shown without EC for illustrative purposes. As can be seen, more than half of the received frame is missing despite the fact that 68% of packets arrive on time. In particular, slices 2, 4, and 7 could not be reconstructed even though most of the packets for these slices were properly received. This is because the few initial packets of those respective slices contained slice headers; the header information within each slice or NAL unit carries a high degree of importance for proper video reconstruction. Therefore, DRAS.264 was designed to exploit this characteristic to increase the chances of proper video reconstruction under less-than-desirable network conditions.

III. Related Work

Prioritized MAC layer retransmission strategies for video streaming were first studied in [9], where retry limits were dynamically assigned to the layers of video encoded by the fine granularity scalability technique of MPEG-4. However, the study ignored playout deadlines, which is undesirable for real-time video. More advanced methodologies were presented in

[10]–[12]. In these methods, optimal retry limits are computed offline for different channel conditions and playout constraints. However, the authors resort to heuristic-based approaches to comply with real-time video constraints. These schemes rely heavily on determining the importance of each packet, which is typically done prior to the encoding process. In [13], packets are discarded based on priority-computed deadlines rather than the retry limit set by the 802.11 MAC protocol.

Cross-layer approaches in streaming H.264 videos are known to leverage 802.11e access categories (ACs) to optimize video delivery. Some methods simply mark parts of the H.264 bitstream, which are then mapped to ACs of varying priority [14]. There also exist techniques that employ automatic repeat request. Some work with offline computations that quantify the impact of error propagation of H.264 bitstreams, while others apply buffer management, channel estimation, and transcoding for improved real-time video delivery [15]–[16]. These are orthogonal to DRAS.264.

The body of work on adaptive retransmission shows general trends of improvement for real-time video streaming. However, earlier studies mostly focus on the efficiency of pre-processing compressed video. This is not necessary for DRAS.264. Furthermore, instead of CIF (352 × 288) video, this work studies HD (1,920 × 1,080) resolution, which is a significant expansion in the scope of research. Finally, no work has studied the effect of 802.11 retry-limit adaptation for H.264 encoded video.

IV. DRAS.264

DRAS.264 is a sender-based scheme consisting of the following two main mechanisms: slice protection (SP) and RA. SP focuses on how packets containing slice headers are transmitted. RA is the process by which retry limits are assigned according to the type of video frame that is being transmitted. Figure 3 shows the architecture of the DRAS.264 scheme, which operates primarily at the MAC layer. The SP Module provides cross-layer communication between the MAC and Network layers and is responsible for processing the drop notification of MAC layer frames that contain slice headers. The RA Module reads the current MAC frame and monitors the 802.11 wireless performance to adaptively adjust the retry limit. It also requires as input the number of packets contained in the current slice being processed, which is communicated from the Application layer. Note that this implementation does not require explicit feedback from the receiving STA.

DRAS.264 closely monitors and parses all 802.11 frames that are being transmitted at the MAC layer. The relative importance of each frame is extracted, yielding four main priority levels — slice headers, I-frames, P-frames, and B-frames (in decreasing order). Furthermore, all packet delays are

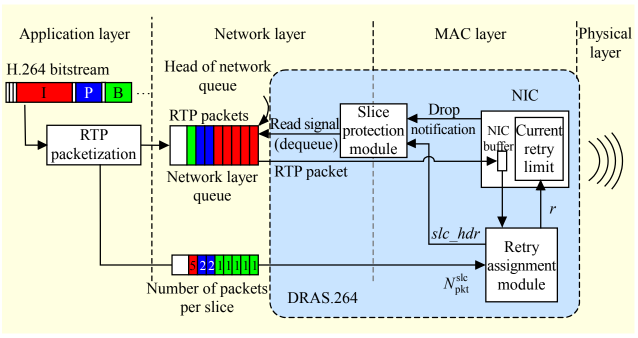


Fig. 3. DRAS.264 architecture.

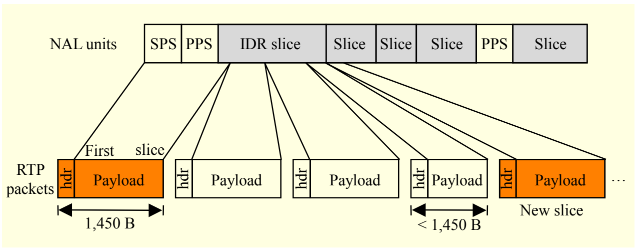


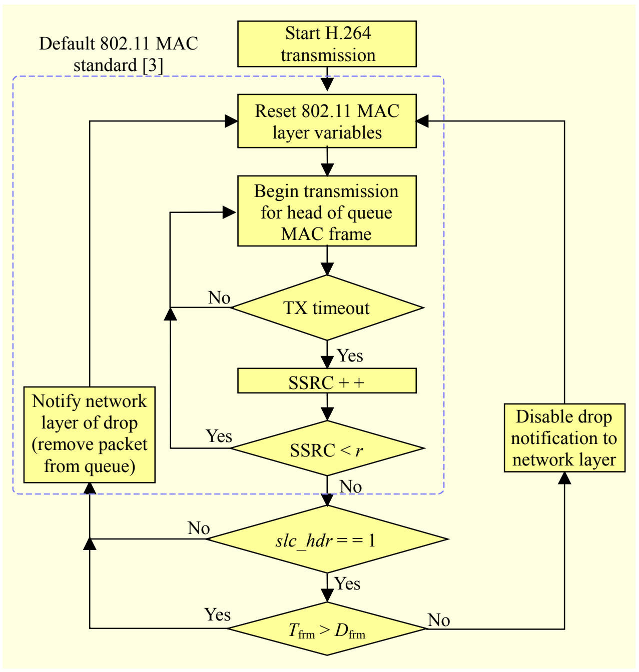
Fig. 4. IETF RFC 6184 merging Sequence Parameter Set (SPS) and Picture Parameter Set (PPS) with slice headers.

monitored to decide on termination of retransmissions for a particular packet if its deadline is expected to be exceeded. Both the relative packet importance and real-time delay information dictate the RA for each packet.

DRAS.264 is based on the Real-time Transport Protocol (RTP) and is able to detect H.264/RTP packets encapsulated in MAC layer frames. The Fragmentation Unit (FU) structure of IETF RFC 6,184 [17] is applied for encapsulation of H.264 data into RTP packets, as outlined in Fig. 4. The detection of H.264/RTP packets involves examining the MAC layer frames residing at the head of the transmission queue of the network interface card (NIC). These frames are parsed to extract the encapsulated RTP packets, which are then further examined for particular RTP header fields. RTP packetization facilitates the detection of new H.264 slices with the marker (*m*) bit in the RTP header. During packetization, *m*-bit is set when a slice is fragmented across multiple packets. At the MAC layer, the value of *m*-bit for the previous packet is stored in a variable, *prev_m*. When *prev_m* = 0, this indicates that the current packet holds information residing at the start of a slice, and hence represents a slice header. The slice type is then saved for use in subsequent steps of the RA process.

1. SP

As discussed in Section II, slice headers are crucial for video reconstruction. Thus, one of the important features of DRAS.264 is the SP mechanism shown in Fig. 5. Note that the default



Default 802.11 MAC standard [3]
Start H.264 transmission
Reset 802.11 MAC layer variables
Begin transmission for head of queue MAC frame
TX timeout
No
Yes
SSRC + +
SSRC < r
Yes
No
Notify network layer of drop (remove packet from queue)
Disable drop notification to network layer
slc_hdr = = 1
No
Yes
T_frm > D_frm
Yes
No

Fig. 5. Flowchart for slice protection.

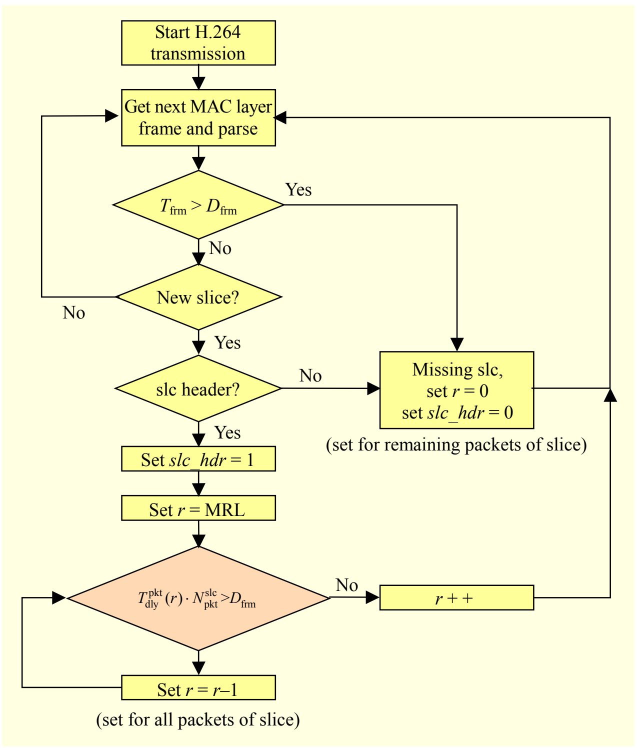
802.11 MAC protocol is still applied. In particular, the STA Short Retry Count (SSRC), which is a MAC layer variable used to count the number of retransmissions, is monitored and checked against the assigned retry limit, r. Only when an H.264 slice header is contained within the packet under transmission ($slc\_hdr = 1$) will the proposed deviate from the default operation. In this case, if r = SSRC, then the packet drop notification to the network layer is blocked. This is subject to the time constraint that the projected packet delivery time, T_{frm}, should not be greater than the frame deadline, D_{frm}, where D_{frm} is computed as

$$D_{\text{frm}} = T_{\text{ini}} + \lambda \times \text{TS}, \tag{3}$$

where T_{ini} is the initial startup delay, λ = 1/fps, and TS is the RTP timestamp. Keep in mind that throughout this process the normal MAC layer operation continues; that is, when SSRC = r, the CW and BO variables are reset, which is represented by the first block in the flowchart. However, since the drop notification to the network layer is disabled for MAC frames holding slice headers, the same packet residing at the head of the network layer queue is allowed another set of retransmissions with the newly set MAC layer variables. Thus, as long as a MAC frame containing a slice header is within its deadline, DRAS.264 will not drop it. However, if $T_{\text{frm}} > D_{\text{frm}}$, then the network layer drop notification is enabled when the assigned retry limit is reached.

2. RA

The second important feature of DRAS.264 is the process by which retry limits are assigned. This task is outlined in Fig. 6. As presented in the figure, the first step after parsing an RTP packet is to compare its T_{frm} with D_{frm} to determine if the current packet has a chance to arrive on time. If this criterion is not met, then the retry limit of the current packet and all subsequent packets of the frame are set to zero. This effectively "drops" packets that are projected to be delayed beyond their deadlines and avoids unnecessary retransmissions. For real-time video, this buys time for subsequent frames and GOP sequences, which by nature of the algorithm will be allocated to higher priority packets.



Start H.264 transmission
Get next MAC layer frame and parse
T_frm > D_frm
Yes
No
New slice?
No
Yes
slc header?
No
Yes
Missing slc, set r = 0 set slc_hdr = 0
(set for remaining packets of slice)
Set slc_hdr = 1
Set r = MRL
T^pkt_dly (r) · N^slc_pkt > D_frm
No
r + +
Set r = r–1
(set for all packets of slice)

Fig. 6. Flowchart for RA.

RA performs on groups of packets rather than on a per-packet basis, and works alongside SP. Specifically, all the packets of a slice are considered as a group and RA only takes place when the first packet of a slice (that is, a slice header) is encountered. Based on the packetization scheme shown in Fig. 4, this has the advantage of protecting SPS and PPS in addition to slice headers. When a slice header is found by way of the m-bit, explained earlier, a slice header flag, slc_hdr, is set and communicated with the SP Module as presented in Fig. 3. This is followed by setting r to the maximum retry limit (MRL), which is the maximum number of retransmissions permitted by the MAC layer. For this work, the default value for MRL is 7.

The predicted packet delay for the rth retry attempt, $T^{\text{pkt}}_{\text{dly}}(r)$, needs to be known. Details on the delay prediction are provided in the next subsection. All the predicted packet delays are kept as an array indexed by r. This is multiplied by the number of packets contained in the slice, $N^{\text{slc}}_{\text{pkt}}$, where "slc" represents the slice type. Thus, $N^{\text{I}}_{\text{pkt}}$, $N^{\text{P}}_{\text{pkt}}$, and $N^{\text{B}}_{\text{pkt}}$ represent the number of packets contained in I-slice, P-slice, and B-slice, respectively. Since this information is readily available during the packetization process, the number of packets per slice is passed down from the application layer to the MAC layer using cross-layer communication, as shown in Fig. 3.

The modification of the retry limit is based on the product $T^{\text{pkt}}_{\text{dly}}(r) \cdot N^{\text{slc}}_{\text{pkt}}$ and the frame deadline, D_{frm}. If the resulting cumulative delay exceeds the frame deadline, then the retry limit is decremented by 1. Otherwise, r holds the last assigned value and the RA module waits for the next MAC layer frame to become available for processing. One important factor during RA is frame sizes. Due to the fact that reference frames typically hold a larger amount of data than non-reference frames (that is, $N^{\text{I}}_{\text{pkt}} > N^{\text{P}}_{\text{pkt}} > N^{\text{B}}_{\text{pkt}}$), directly comparing D_{frm} against $T^{\text{pkt}}_{\text{dly}}(r) \cdot N^{\text{slc}}_{\text{pkt}}$ would give the most important packets the least amount of time for transmission and thus the lowest retry limits. This is contradictory to the relative importance of each slice type within an H.264 bitstream. Therefore, the time computed per packet, $D_{\text{frm}}/N^{\text{slc}}_{\text{pkt}}$, is exchanged between B-slices and I-slices. That is, $N^{\text{B}}_{\text{pkt}}$ is used for I-slices, while $N^{\text{I}}_{\text{pkt}}$ is used for B-slices for the inequality in Fig. 6. The computation for P-slices remains unchanged with $N^{\text{P}}_{\text{pkt}}$. This ensures the selection of retry limits is proportional to the impact those packets will have on the received video.

3. Prediction of Packet Delays

Recall from Section II that a packet can be in one of the following four states: success, collision, BO, and deferred. Based on these states, an analytical model is derived to predict packet delays. For an STA employing the DRAS.264 scheme, a running average of the time spent per state is recorded at each transmission attempt, and $T^{\text{pkt}}_{\text{dly}}(r)$ is made up of the sum of those average delays. The main factors contributing to the overall delay between when a packet is transmitted to the time the corresponding ACK frame is successfully received are as follows:

- n_{c}: collisions;
- BO_r: BO variable for rth retry attempt; and
- n_{tx}: number of deferrals before ACK is received.

Based on these metrics, $T^{\text{pkt}}_{\text{dly}}(r)$ is given by the following equation:

$$T^{\text{pkt}}_{\text{dly}}(r) = \sum_{r=0}^{n_{\text{c}}} T_{\text{bkf}}(r) + n_{\text{c}} T_{\text{col}} + n_{\text{tx}} T_{\text{bsy}} + T_{\text{suc}} + (n_{\text{c}} + n_{\text{tx}}) T_{\text{DIFS}}, \quad (4)$$

where $T_{\text{bkf}}(r)$ represents the delay associated with the BO state for the rth retry, and T_{bkf}, T_{col}, T_{bsy}, and T_{suc} represent the time spent in "BO," "collision," "busy," and "successful" states, respectively. These terms are defined as follows:

- $T_{\text{bkf}}(r) = \text{BO}_r \times T_{\text{slot}}$, where $\text{BO}_r \in [0, \text{CW}_r]$ and $\text{CW}_r = a2^r - 1$, where $a = \text{CW}_{\text{init}} + 1$. Note that CW_{init} differs from one standard to another (such as 802.11a and 802.11b).
- T_{col} and T_{bsy} refer to the time consumed during a collision or deferral state, respectively. Since the underlying assumption in this work considers all MAC layer frames to be of equal length, both parameters have the same computation given in (1): $T_{\text{bsy}} = T_{\text{col}} = T_{\text{frm}} + T_{\text{SIFS}} + T_{\text{ACK}}$, where T_{frm}, T_{SFIS}, and T_{ACK} are given in Table 1.
- T_{suc} is the duration for a successful transmission, and is equivalent to T_{bsy} due to the aforementioned assumption regarding the MAC frame size.

Since $T_{\text{bsy}} = T_{\text{col}}$, (4) can be reformulated as

$$T^{\text{pkt}}_{\text{dly}} = \left[T_{\text{bkf}}(0) + T_{\text{suc}}\right] + \left[\sum_{r=1}^{n_{\text{c}}} T_{\text{bkf}}(r)\right] + \left[\left(n_{\text{c}} + n_{\text{tx}}\right)\left(T_{\text{bsy}} + T_{\text{DIFS}}\right)\right], \quad (5)$$

where terms enclosed in square brackets represent specific delays associated with the four main states. The first bracketed computation refers to the time spent during a successful transmission; the second bracket refers to the total time spent in the BO state; and the third term represents the time spent during the collision and deferral states.

For the default 802.11 operation, a MAC layer frame may absorb up to six retransmissions. Therefore, a certain delay is incurred per retry attempt, which only varies based on the BO counter and deferral time. For the basis of prediction, the average BO window size per attempt, $\text{E}[\text{BO}_k]$, and the average number of transmissions other STAs gain access to the medium for the kth attempt, $\text{E}[n_{\text{tx}_k}]$, are recorded during transmissions. Also, considering the case for a successfully transmitted frame at the rth attempt, the number of collisions that take place during the transmission is equal to the number of retry attempts; that is, $n_{\text{c}} = r \le R$. Note that R represents the maximum number of retries allowed, which is six for the default MAC protocol. The components $\text{E}[\text{BO}_k]$, $\text{E}[n_{\text{tx}_k}]$, and the replacement of n_{c} by r lead to the following general equation for the estimation of packet delay based on the number of retry attempts, $T^{\text{pkt}}_{\text{dly}}(r)$:

$$T^{\text{pkt}}_{\text{dly}}(r) = T_{\text{suc}} + \sum_{k=0}^{r} \text{E}\left[\text{BO}_k\right] T_{\text{slot}} + r\left(T_{\text{col}} + T_{\text{DIFS}}\right) + \sum_{k=0}^{r} \text{E}[n_{\text{tx}_k}]\left(T_{\text{bsy}} + T_{\text{DIFS}}\right). \quad (6)$$

Equation (6) is realized as an array of predicted delays indexed by the retry attempt r. The running averages of BO times, $\text{E}[\text{BO}_k]$, and instances of deferral, $\text{E}[n_{\text{tx}_k}]$, are summed for all attempts up to and including the retry attempt in question to account for the delays from the previous attempts.

The delays computed in (6) tend to under-predict the actual end-to-end delays experienced at the MAC layer. Under-prediction does not pose an issue when assigning retry limits to packets, because lower per-packet delays constitute assigning retry limits greater than the minimum value needed to successfully deliver a packet. However, if over-prediction takes place, a retry limit that is too low for successful delivery will be selected. Thus, DRAS.264 employs some leniency in the RA process by assigning one higher value to *r*, as shown in the last step of Fig. 6.

4. Bandwidth Availability Estimation

One drawback to the originally proposed DRAS.264 algorithm in [5] is that retry limits would be unnecessarily reduced, regardless of the video streaming performance. Thus, for stretches of video, the default MAC layer operation would outperform DRAS.264 at times. Therefore, a method to conditionally apply DRAS.264 was cited as a necessary component for better performance. To implement this, the RA Module of DRAS.264 is augmented with bandwidth estimation to determine when to activate/deactivate the RA process. Bandwidth availability estimation in DRAS.264 considers the maximum MAC layer packet size and inter-ACK arrival times to calculate the *goodput*. This is done by keeping track of ACK arrival times, t_{ACK}, during transmissions. When a new ACK arrives, the inter-ACK time is computed using $T_{\Delta(\text{ACK})_i} = t_{\text{ACK}(i)} - t_{\text{ACK}(i-1)}$, where $t_{\text{ACK}(i)}$ and $t_{\text{ACK}(i-1)}$ are the ACK arrival times for the two previous successfully delivered MAC layer frames.

The maximum frame size is divided by the most recently computed inter-ACK time, $T_{\Delta(\text{ACK})_i}$, to obtain the goodput, where the MAC frame size including headers amounts to 1,506 bytes and is referred to as *MAC_frame_size*. Thus, the following equation can be used to represent the effective available bandwidth BW_i:

$$\text{BW}_i = \text{MAC\_frame\_size} / T_{\Delta(\text{ACK})_i}. \tag{7}$$

Equation (7) gives an instantaneous measure of network performance. However, it is important to capture the trend over time to avoid unnecessarily activating RA in DRAS.264. Thus, an exponentially weighted moving average is applied to BW_i, which is given as

$$\text{avg\_BW}_i = \alpha \times \text{BW}_i + (1 - \alpha) \times \text{avg\_BW}_{i-1}, \tag{8}$$

where α is a weighting factor. During streaming, avg_BW_i is monitored and compared against a predefined threshold, BW_th. If avg_BW_i falls below the threshold, then DRAS.264 is activated and retry adjustment operates as described in Fig. 5. Otherwise, retry limits revert to the default value and remain static. The selection of an optimal α and BW_th is beyond the scope of this paper. However, favorable results have been observed with $\alpha = 0.2$ and BW_th in the range of 10 Mbps to 26 Mbps for different scenarios in a 54 Mbps network.

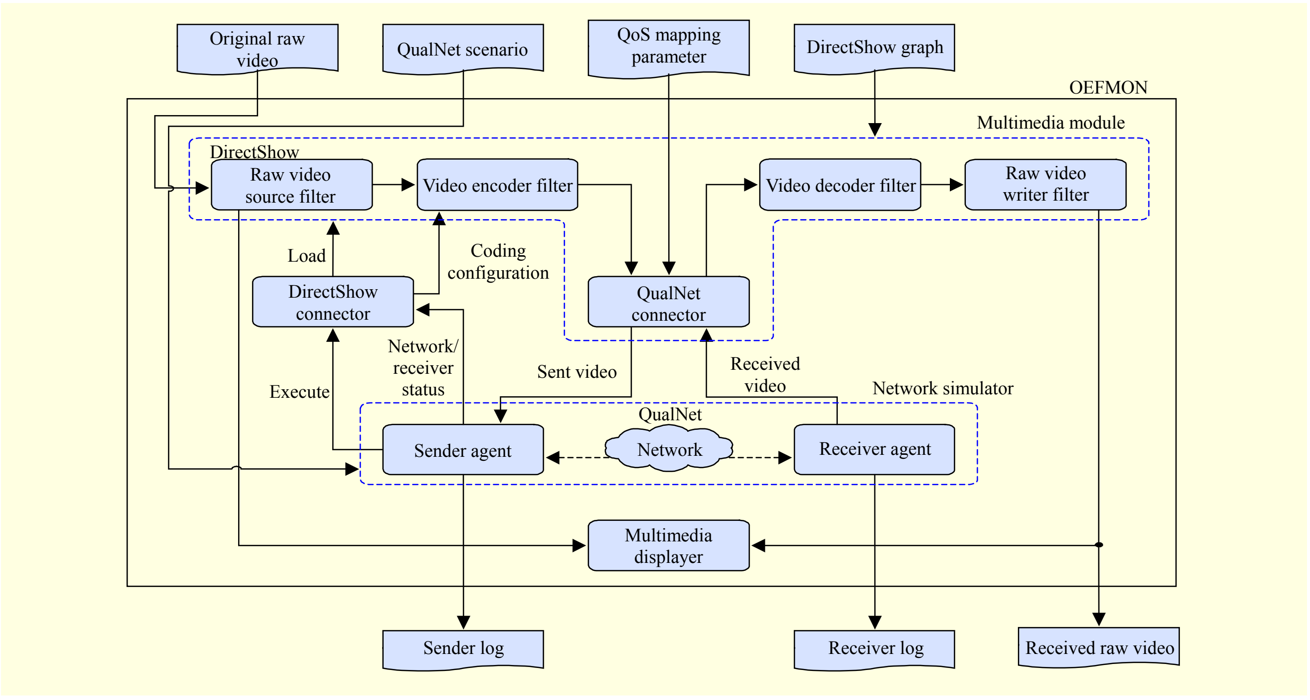


Fig. 7. OEFMON simulation environment.

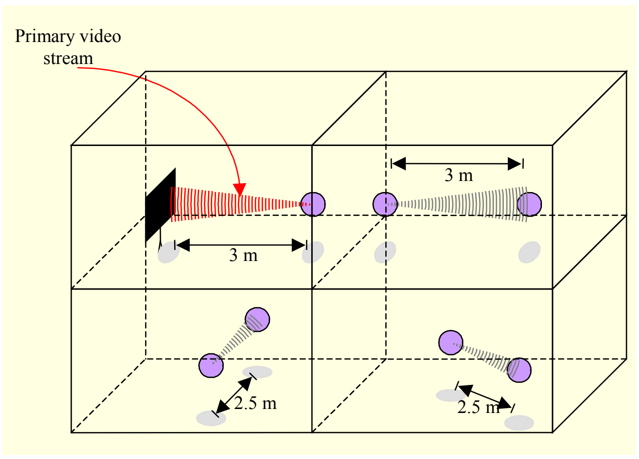


Fig. 8. Experimental setup.

V. Simulation Study

Our simulation study was performed on within the Open Evaluation Framework for Video Over Networks (OEFMON) designed at the Korea Advanced Institute of Science and Technology [18]. Figure 7 shows the OEFMON framework, which integrates DirectShow, a multimedia engine, with the QualNet network simulator [19]. In our work, the raw video source input shown in the figure is bypassed, and a pre-encoded H.264 video is directly sent via the network to accurately portray contemporary in-home streaming demands. Furthermore, OEFMON has been updated to allow for multi-slice video simulations, a main feature of the H.264 codec designed for lossy network environments.

To simulate a practical local streaming experience, eight nodes are arranged as depicted in Fig. 8, representing a typical apartment scenario. Four simultaneous streams exist among the eight nodes, the primary stream being shown in red. All the results obtained are for the primary stream whose sender is DRAS.264-enabled. The remaining three streams are used to inject background traffic at a constant bit rate (CBR) using the default IEEE 802.11 MAC protocol. Streaming distances are less than or equal to 3 m, which is a suitable viewing range for home networks (see Fig. 8). Although this range may vary, for the given scenario it does not impact performance. In other words, as long as transmitting nodes are within the carrier sense range of each other (up to 100 m), interference will exist amongst video streams.

Three short video clips (Sony Bravia, African Cats, and LG Racing) were used for the primary video stream in Fig. 8. The clips were encoded using the widely known open source encoder x264 [20] with the Main Profile, Level 4.1 at 1,080p and 30 fps. African Cats and LG Racing were encoded using eight slices and with average bitrates of 4 Mbps and 5 Mbps, respectively. Sony Bravia was encoded using four slices with an average bitrate of 7.2 Mbps. The test clips range in length from 315 frames to 450 frames. In addition, RTP over UDP was used for the streaming protocol. The network scenario was created in QualNet version 5.0.2 with an 802.11a network and a bandwidth of 54 Mbps. Two background scenarios were considered — high congestion and burst congestion. For high congestion, all three background streams are CBR traffic at 10 Mbps and this was tested for both 8-slice videos (African Cats and LG Racing). The background streams start one second after the primary video stream is initiated and continue until the end of simulation. Burst congestion is tested with the Sony Bravia clip, and the background traffic is gradually increased starting at 2 s with 10 Mbps, then an additional 5 Mbps background stream introduced at 3 s, followed by a third 5 Mbps stream introduced at 4 s. The burst ends at 5 s.

Figure 9 shows the peak signal-to-noise ratio (PSNR) results for the received videos in reference to the original undistorted video for DRAS.264 (red lines) and the default 802.11 MAC protocol (blue lines). Since a PSNR value of 37 dB (green dashed line) is considered "excellent" quality [21], any values above 40 dB are saturated. The estimated bandwidth avg_BW computed by (8) is also shown on the graphs (black lines) with the selected bandwidth threshold, BW_th, indicated by a red dashed line. When avg_BW < BW_th, the retry limit modification process of DRAS.264 takes effect (see Fig. 6). Otherwise, the retry limit is set to the default static value.

One important observation from the simulation results is that virtually no packet loss occurs for the default 802.11 MAC layer (except for one packet from African Cats). Thus, any PSNR degradation seen for the default 802.11 MAC in Fig. 9 is due to delivery of packets beyond their playout deadlines. These are wasted transmissions that accumulate delay and lead to progressive degradation due to error propagation. For Sony Bravia, African Cats, and LG Racing, the percentages of packets that miss their playout deadlines are 2.2%, 23.7%, and 51.0%, respectively. The low percentage for Sony Bravia is because it is experiencing burst congestion. Thus, Fig. 9(a) focuses on the part of the graph where the burst occurs, between frames 30 and 120. All other parts of the video exhibit perfect PSNR.

Figures 9(b) and 9(c) present results for video streamed under high congestion conditions where background traffic is continuous. Nevertheless, DRAS.264 is able to lessen the impact of interference. In contrast, the default 802.11 MAC operation has difficulty recovering from the accumulated delay.

With DRAS.264, the percentages of total packets that miss their playout deadlines drop to 1.6%, 18.7%, and 23.0% for Figs. 9(a), 9(b), and 9(c), respectively. When factoring in the true packet drops, the percentages are 2%, 20%, and 31.9%, respectively. This is due to two reasons — reducing retry limits

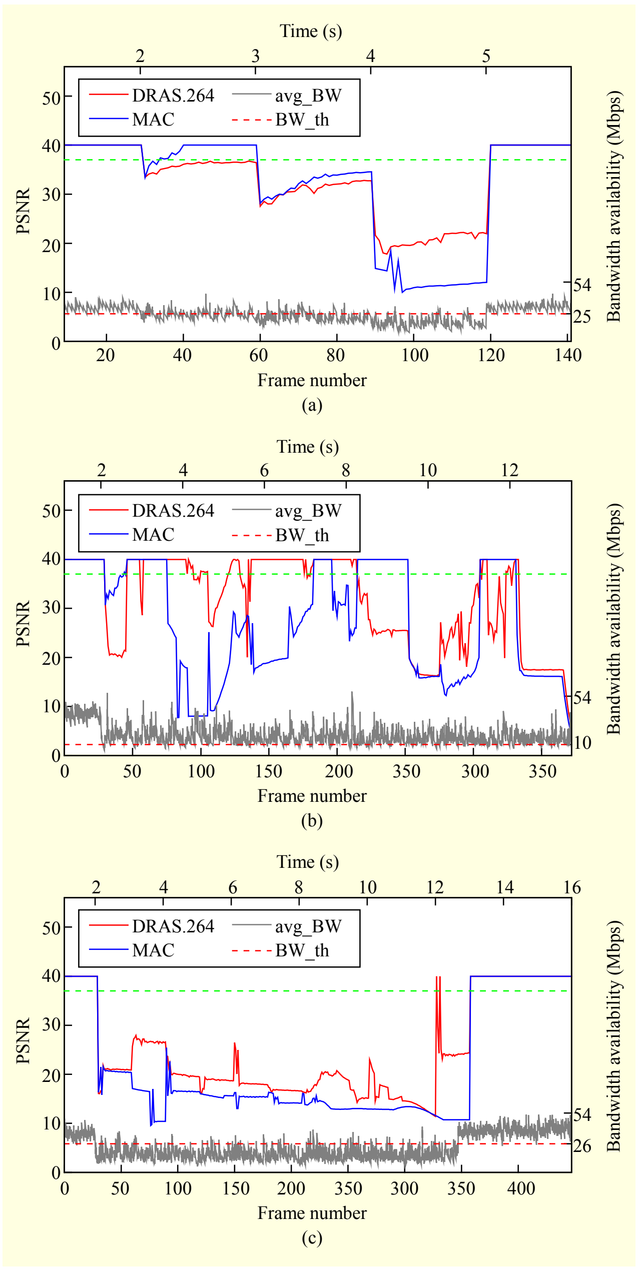


Fig. 9. PSNR vs. frame number: (a) Sony Bravia (burst congestion), (b) African Cats (high congestion), and (c) LG Racing (high congestion).

and dropping expired packets. However, DRAS.264 still has a lower effective packet loss rate than the default 802.11 MAC protocol. Furthermore, the average PSNR results improve by 0.39 dB for Sony Bravia, 4.45 dB for African Cats, and 3.48 dB for LG Racing.

Figure 10 shows the resulting visual improvements when DRAS.264 is applied to video streaming. Frame number 115 (a B-frame) of Sony Bravia is presented in Fig. 10(a). Note that the default MAC 802.11 protocol (the image on the right) suffers heavily from error propagation. The green colors in the

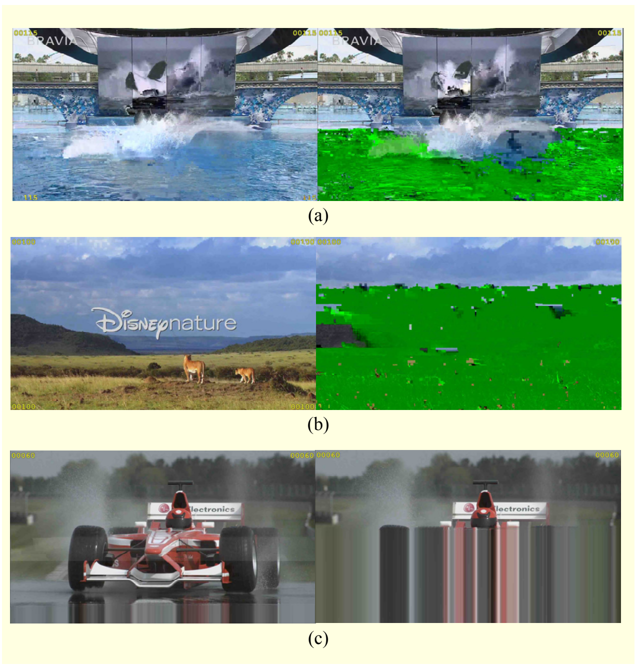


Fig. 10. Visual comparison of DRAS.264 and default 802.11 MAC operation. Images on left are result of DRAS.264 streaming and images on right are based on default 802.11 protocol: (a) frame 115 of Sony Bravia (B-frame), (b) frame 100 of African Cats (P-frame), and (c) frame 60 of LG Racing (I-frame).

frame indicate that the decoder is attempting to reference information that is completely lost. This can be traced back to the PSNR graph in Fig. 9(a). Specifically, the PSNR improvement for DRAS.264 over the default MAC 802.11 protocol is observed during the GOP containing frames 90 to 119. The severe error propagation seen for the default streaming method comes from an early P-frame in the GOP (frame 93) that missed its playout deadline. During this frame sequence, the heaviest burst of congestion takes place. Thus, avg_BW is below BW_th (25 Mbps) for the majority of the time between 4 s and 5 s, and DRAS.264 is enabled. During the operation of DRAS.264, retry limits for B-slices are reduced, allocating more time for slices of a higher priority. This allows frame 93 to be delivered, reducing the observed error propagation.

Figure 10(b) presents frame 100 (a P-frame) from the African Cats clip, where the observed distortion is similar to that discussed for Fig. 10(a). For the default MAC operation, the error propagation seen in this frame originates from unsuccessful delivery of multiple reference frames early in the GOP sequence that spans frames 76 to 105. Although the IDR frame of the GOP (frame 76) is mostly received in tact, frames 77–82 miss their deadlines leading to very poor quality throughout the remainder of the GOP sequence. This can be seen in the PSNR graph in Fig. 9(b) for the default MAC operation. Note that a sharp drop takes place just after frame 76 and the PSNR values remain below 20 dB until the next IDR frame (frame 106) is encountered.

Nevertheless, DRAS.264 is still able to maintain high PSNR values during the same GOP sequence and virtually no distortion is noticeable in frame 100. The slight reduction seen on the PSNR graph for frame 100 is difficult to find on the actual image. However, the distortion is contained in the first slice and originates from dropped packets in frame 94. Thus, subsequent top slices reference incorrect macroblocks, leading to minor error propagation.

Note that BW_th is set to a very low value of 10 Mbps in Fig. 9(b). This decision can be attributed to the lower bitrate (4 Mbps) used to encode the African Cats video, which generates less data. Longer inter-ACK arrival times can be mistaken for poor network conditions. Therefore, in this particular scenario, setting BW_th to lower values avoids activating the DRAS.264 scheme when it is unnecessary to do so.

Finally, Fig. 10(c) shows frame 60 (an I-frame) from the LG Racing clip. For DRAS.264, the last packet of slice 4 is lost. Note that this has a slight effect on the reconstruction of slice 5 because some inter-dependencies exist between slices. For the last slice, only the first packet is delivered on time leading to weighted pixel average (WPA) being applied to the remainder of the slice. The default MAC protocol suffers a more severe degradation on frame 60. Slices 5–8 and part of slice 4 are delivered beyond their deadlines, which are then considered lost by the decoder leading to WPA being used for over half the frame.

Since frame 60 is an I-frame, it serves as the main reference frame for the subsequent frames in the GOP. Therefore, as seen in Fig. 9(c), PSNR for DRAS.264 remains above 25 dB while PSNR for the default MAC protocol is below 17 dB throughout the GOP. Also, many frames are lost for the default MAC case, most notably frames 61–75, frames 126–140, and frames 229–357. When frames are lost, the last frame in the display buffer is repeated, which can be visually described as a pause in playback. Once a frame is properly delivered, the video makes an abrupt transition to the current scene being streamed. During the stretch of high congestion that starts at frame 30 and ends at frame 350, DRAS.264 mostly avoids this phenomenon and is able to provide a smoother video playback due to less frame loss.

VI. Conclusion and Future Work

This paper presented DRAS.264, which is an adaptive MAC layer retransmission scheme tailored to H.264 videos. DRAS.264 also incorporates a bandwidth estimation method to control the retry adjustment process and cross-layer communication to provide the exact number of packets per slice aiding in more accurate RAs per packet. Our simulation study using OEFMON shows that DRAS.264 results in better overall video quality when compared to the static retry limit scheme of the default 802.11 MAC layer.

DRAS.264 can be further improved in two areas. First, accessing packets in the network layer queue and purging those packets that are expected to exceed their deadlines can help to further reduce unnecessary delay. Second, dynamic selection of α for the bandwidth estimation process as well as finding an optimal bandwidth threshold have a strong potential to fine-tune DRAS.264 activation for improved performance.

References

[1] Intel, *Intel WiDi® and Intel Pro® Wireless Display*, Intel Coporation©, 2014. Accessed July 8, 2014. http://www.intel. com/content/www/us/en/architecture-and-technology/intel-wireless-display.html

[2] Apple, *Airplay–play content from iOS on Apple TV*, Apple Inc. ©, 2014. Accessed July 8, 2014. https://www.apple.com/airplay/

[3] IEEE Std. 802.11™-2012, *IEEE Standard for Inform. Technol. Part 11: Wireless LAN Medium Access Contr. (MAC) and Physical Layer (PHY) Specifications*, IEEE, Piscataway, NJ, USA, 2012.

[4] J. Greengrass, J. Evans, and A. Begen, "Not All Packets are Equal, Part 2: The Impact of Network Packet Loss on Video Quality," *IEEE Internet Comput*., vol. 13, no. 2, Mar. 2009, pp. 74–82.

[5] M. Sinky et al., "DRAS.264: A Dynamic Retry Adaptation Scheme to Improve Transmission of H.264 HD Video over 802.11 Peer-to-Peer Networks," *ICUIMC*, Siem Reap, Cambodia, no. 51, Jan. 9–11, 2014, pp. 1–8.

[6] ITU-T Rec. H.264 | ISO/IEC 14496-10, *AVC: Advanced Video Coding for Generic Audiovisual Services*, Feb. 2014.

[7] T. Stockhammer, M. Hannuksela, and T. Wiegand, "H.264/AVC in Wireless Environments," *IEEE Trans. Circuits Syst. Video Technol*., vol. 13, no. 7, July 2003, pp. 657–673.

[8] I. Richardson, "*The H.264 Advanced Video Compression Standard*," West Sussex, UK: John Wiley and Sons, 2010, pp. 237–248.

[9] Q. Li and M. van der Schaar, "Providing Adaptive QoS to Layered Video over Wireless Local Area Networks through Real-Time Retry Limit Adaptation," *IEEE Trans. Multimedia*., vol. 6, no. 2, Apr. 2004, pp. 278–290.

[10] M. van der Schaar, D. Turaga, and R. Wong, "Classification-Based System for Cross-Layer Optimized Wireless Video Transmission," *IEEE Trans. Multimedia*., vol. 8, no. 5, Oct. 2006, pp. 1082–1095.

[11] M. van der Schaar and D. Turaga, "Cross-Layer Packetization and Retransmission Strategies for Delay-Sensitive Wireless Multimedia Transmission," *IEEE Trans. Multimedia*., vol. 9, no.

1, Jan. 2007, pp. 185–197.

[12] C.-M. Chen, C.-W. Lin, and Y.-C. Chen, "Cross-Layer Packet Retry Limit Adaptation for Video Transport over Wireless LANs," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 20, no. 11, Nov. 2010, pp. 1448–1461.

[13] M. Lu, P. Steenkiste, and T. Chen, "A Time-Based Adaptive Retry Strategy for Video Streaming in 802.11 WLANs," *Wireless Commun. Mobile Comput.*, vol. 7, no. 2, Feb. 2007, pp. 187–203.

[14] A. Ksentini, M. Naimi, and A. Gueroui, "Toward an Improvement of H.264 Video Transmission over IEEE 802.11e through a Cross-Layer Architecture," *IEEE Commun. Mag.*, vol. 44, no. 1, Jan. 2006, pp. 107–114.

[15] P. Bucciol et al., "Cross-Layer Perceptual ARQ for H.264 Video Streaming over 802.11 Wireless Networks," *IEEE GLOBECOM*, Dallas, TX, USA, Nov. 29–Dec. 3, 2004, vol. 5, pp. 3027–3031.

[16] A. Moid and A. Fapojuwo, "A Cross-Layer Framework for Efficient Streaming of H.264 Video over IEEE 802.11 Networks," *J. Comput. Syst. Netw. Commun.*, vol. 2009, Apr. 2009, pp. 1–13.

[17] IETF 6184, "RTP Payload Format for H.264 Video," May 2011.

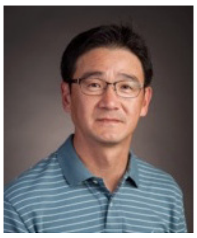
[18] C. Lee et al., "OEFMON: An Open Evaluation Framework for Multimedia over Networks," *IEEE Commun. Mag.*, vol. 49, no. 9, Sept. 2011, pp. 153–161.

[19] Scalable Network Technologies, Inc., "QualNet 5.0.2 User's Guide," 2010.

[20] VideoLAN Organization, *x264, the Best H.264/AVC Encoder*, 2014. Accessed July 9, 2014. http://www.videolan.org/developers/x264.html

[21] J. Gross et al., "Cross-Layer Optimization of OFDM Transmission Systems for MPEG-4 Video Streaming," *Comput. Commun.*, vol. 27, no. 11, July 2004, pp. 1044–1155.

Mohammed Sinky received his BS and MS degrees in computer engineering from Oregon State University (OSU), Corvallis, USA, in 2001 and 2004, respectively. He worked as a lecturer at Umm Al-Qura University, Makkah, Saudi Arabia, between 2005 and 2007. He returned to OSU and earned his PhD degree in electrical and computer engineering, in 2015. His interests include embedded systems, video compression, wireless video streaming, and parallel processing.



Ben Lee received his BE degree in electrical engineering in 1984 from the Department of Electrical Engineering, State University of New York, Stony Brook, USA and his PhD degree in computer engineering in 1991 from the Department of Electrical and Computer Engineering, Pennsylvania State University, University Park, USA. He is currently a professor at the School of Electrical Engineering and Computer Science, Oregon State University (OSU), Corvallis, USA. He has published over 100 conference proceedings, book chapters, and journal articles in the areas of embedded systems; computer architecture; multithreading and thread-level speculation; parallel and distributed systems; and wireless networks. He received the Loyd Carter Award for Outstanding and Inspirational Teaching and the Alumni Professor Award for Outstanding Contribution to the College and the University from the OSU College of Engineering in 1994 and 2005, respectively. He also received the HKN Innovative Teaching Award from Eta Kappa Nu, School of Electrical Engineering and Computer Science, in 2008. He has been on program and organizing committees for numerous international conferences, including 2003 International Conference on Parallel and Distributed Computing Systems, 2005-2011 IEEE Workshop on Pervasive Wireless Networking, 2006, 2007, and 2009, and IEEE International Conference on Pervasive Computing and Communications. He was also a keynote speaker at the 2014 International Conference on Ubiquitous Information Management and Communication. He is currently the chair for the Social, P2P, and Multimedia Networking, Services and Applications track for the 2016 IEEE Consumer Communications and Networking Conference. His research interests include wireless networks; embedded systems; computer architecture; multithreading and thread-level speculation; and parallel and distributed systems.

Tae-Wook Lee is a chief research engineer at LG Display, Timing Controller Development Team, Paju, Rep. of Korea. He received his BS, MS, and PhD degrees in electrical engineering from the University of Ulsan, Rep. of Korea, in 1998, 2000, and 2004, respectively. His PhD work was on inner product optimization and its application to image compression. In 2004, he joined LG Display Co., and since then has worked on the backlight driving, LED local dimming, and timing controllers for LCD TVs.

Chang-Gone Kim is a chief research engineer at LG Display, Timing Controller Development Team, Paju, Rep. of Korea. He received his BS and MS degrees in electrical engineering from Kyungpook National University, Daegu, Rep. of Korea. Currently, he is responsible for the design of timing controllers as a leader of the Timing Controller Development Team, LG Display R&D Center.

Jong-Keun Shin received his BS and MBA degrees in electrical engineering from Kyungpook National University, Daegu, Rep. of Korea. He is a vice-president at LG Display, Paju, Rep. of Korea, where he worked on applied technology and circuit development. Currently, he is the CEO of eCONY Co., Ltd., Gumi, Rep. of Korea.