DRAS.264: A Dynamic Retry Adaptation Scheme to Improve Transmission of H.264 HD Video Over 802.11 Peer-to-Peer Networks

Mohammed Sinky and Ben Lee
Oregon State University
Corvallis, Oregon 97331
{sinky, benl}@eecs.orst.edu

Tae-Wook Lee, Chang-Gone Kim, and Jone-Keun Shin
LG Display Co. Ltd., LCD Laboratory
Paju-Si, Gyeonggi-do, Korea
{twlee, cgkim02, rgbshin}@lgdisplay.com

ABSTRACT

Wireless High Definition Video Transmission (WHDVT) over 802.11-based networks enjoys widespread deployment among today's multimedia solutions. Examples include Intel's WiDi^{TM}and Apple's Airplay^{TM}, to name a few. In these systems, peer-to-peer networks are established over which H.264-encoded video is transported wirelessly to be decoded and played back at the receiving node. Built-in reliability at the IEEE 802.11 Medium Access Control (MAC) layer retransmits lost packets up to a default number of retries. Excessive delay induced by such retransmissions can violate the tight playout deadlines for HD content. Furthermore, lower priority packets may be delivered at the expense of delaying other packets of higher visual impact on the displayed video. To mitigate this problem, this paper proposes Dynamic Retry Adaptation Scheme (DRAS.264) tailored to today's compression standard of choice–the H.264/AVC codec. DRAS.264 parses H.264 bitstreams on-the-fly to dynamically adjust retransmission limits at the MAC layer. Simulation results show significant PSNR improvements (over 10 dBs) for stretches of received video under DRAS.264 over the default MAC layer operation.

1. INTRODUCTION

Wireless High Definition Video Transmission (WHDVT) over 802.11-based networks is highly prevalent in today's multimedia solutions. Wide-spread deployment of 802.11 networks in homes makes WHDVT an ideal choice as evident by existing technologies, such as Intel's WiDi [1] and Apple's Airplay [2]. In these systems, a peer-to-peer network is established over which encoded video is wirelessly transmitted between a sender and a receiver. However, significant challenges exist in delivering smooth playback of HD content as WHDVT becomes more pervasive and multiple streams will need to be supported on the same network. Due to the lossy nature of wireless media, the IEEE 802.11 Medium Access Control (MAC) layer incorporates built-in reliability by retransmitting lost packets [4]. At the same time, real-time requirements imposed on packet delivery for video playback may be violated as a result of excessive delay induced by the MAC layer retransmissions. This coupled with the fact that compressed video packets exhibit unequal importance [6], the MAC layer may spend time delivering a lower priority packet at the expense of delaying other packets that have a higher visual impact on displayed video. To mitigate this problem, this paper proposes an adaptive MAC layer retransmission scheme, called *Dynamic Retry Adaptation Scheme* (DRAS.264), tailored to H.264 videos.

The basic idea of the proposed DRAS.264 is to parse H.264 bitstreams in real-time, and then dynamically adjust the MAC layer retry limits according to packet contents, network conditions, and frame playout deadlines. This is achieved by assigning higher retry limits to packets that transport *slice headers* and *slice data for I-frames*, which have higher impact on visual quality. This gives priority to slice headers, without which decoders are unable to reconstruct video frames, and to slice data for I-frames that are needed to prevent error propagations. On the the hand, lower retry limits are assigned to the rest of the bitstream content to compensate for the additional time required to send the more important components, and to prevent transmission of packets that are expected to exceed their respective playout deadlines to avoid unnecessary retransmissions.

This paper is organized as follows. Sec. 2 provides background on the problems associated with contention at the MAC layer and the important features of the H.264 codec. Sec. 3 discusses the related work. The DRAS.264 algorithm is presented in Section 4. Sec. 5 shows a comparison between the performance of DRAS.264 and the default MAC layer operation. Finally, Section 6 concludes the paper and discusses future work.

2. BACKGROUND

In 802.11 networks, access to the shared medium is governed by the *Distributed Coordination Function* (DCF). As with any shared medium, coordination of multiple users essentially serializes transmission of packets resulting in additional end-to-end delay. In the context of real-time video

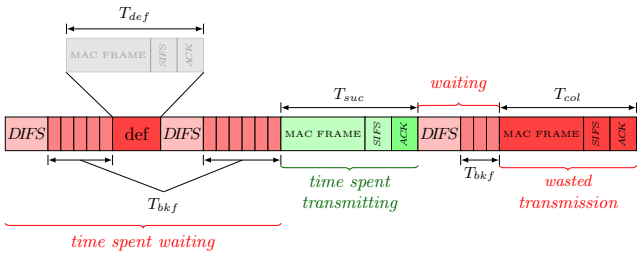


Figure 1: IEEE 802.11 DCF timing.

transmission, this added delay can degrade the quality of received video when playout deadlines are violated. Therefore, it is important to understand the types of delays caused by DCF in order to estimate network conditions pertinent to the operation of DRAS.264. The following subsections discuss these details.

2.1 802.11 DCF Operation

In DCF, each station (STA) monitors the wireless medium for activity in intervals of *time slots* (T_{slot}) [4]. When the medium is sensed to be idle for a specified period of T_{DIFS}, STAs are permitted to transmit. However, to avoid collisions due to simultaneous transmissions from multiple STAs, an exponential backoff mechanism is employed whereby a *random backoff* (BO) time slot is chosen from a *contention window* (CW). A CW is made up of an integral number of time slots, where its initial size depends on the standard, and doubles in size after each successive collision. When the CW reaches a pre-defined maximum value, it remains at this value until a successful transmission takes place or the retry limit for the packet is reached (i.e., dropped).

Fig. 1 shows an example timing of the IEEE 802.11 DCF from the perspective of a STA sharing the medium with other STAs. A STA can be classified to be in one of four states: *success*, *collision*, *backoff*, and *deferred*, which are also indicated in Fig 1. Initially, following a *DIFS* time period, the STA obtains a random backoff value of 11 and starts the countdown process, which represents the *backoff* state. After 5 time slots, another STA is sensed accessing the medium and thus the STA must defer access and transitions to the *deferred* state. When the medium becomes free and a *DIFS* interval has passed, the STA continues in the backoff state for the remaining 6 time slots. The STA then gains access to the medium and transmits without a collision, representing the *success* state. Following a successful transmission, the same process for a new packet begins with a new random backoff value of 3. However, after a *DIFS* interval and a duration of 3 time slots, the STA seizes the medium while another node is simultaneously accessing the medium, putting the STA in the *collision* state.

Table 1 provides the timing calculations for each state under the assumption that all packets are of equal length. Based on this assumption, the *success* (T_{suc}), *collision* (T_{col}), and *deferral* (T_{def}) times are equal. The durations of these states cover the time a sender begins transmission of a packet to the time it *expects* to receive an acknowledgement (ACK). As shown in Table 1, this is the sum of the time required to transmit a MAC layer data frame (T_{frm}), the short inter-

State	Duration
Success	$T_{suc} = T_{frm} + T_{SIFS} + T_{ACK}$
Collision	$T_{col} = T_{frm} + T_{SIFS} + T_{ACK}$
Deferred	$T_{def} = T_{frm} + T_{SIFS} + T_{ACK}$
Backoff	$T_{bkf}(r) = BO_r \times T_{slot} \in (CW_r \times T_{slot})$

Table 1: Duration calculations for the four states of 802.11 DCF.

frame space period (T_{SIFS}), and the time required to transmit an ACK frame (T_{ACK}). The timing calculation for the *backoff* state (T_{bkf}) only depends on the random time slot that is chosen for the r^{th} retry, BO_r.

2.2 The H.264 Codec

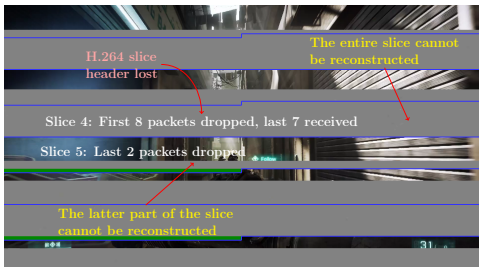
H.264 is a highly efficient coding standard that uses predictive methods to reconstruct video sequences. An encoded video consists of a sequence of *group of pictures* (GOP), which is a set of coded pictures that specifies the order of I-, P-, and B-frames. The interdependencies between frames can lead to error propagation within a GOP sequence when packet loss occurs. Frames can contain a mixture of different macroblock (MB) types, where each MB is a 16×16-pixel region. MBs are labelled according to types of references made for prediction. For example, a B-frame holds MBs that are bi-predicted; however, it may also contain intra-predicted MBs. A P-frame contains MBs that are predicted from past frames and may also contain intra-predicted MBs. I-frames contain only intra-predicted MBs and do not reference other frames.

An HD video frame encoded using H.264 is typically subdivided into multiple *slices*. Slices are classified by the types of MBs they contain. The five slice types supported by the H.264 standard are I (and IDR), P, B, SP, and SI. An IDR or *Instantaneous Decoder Refresh* slice is an I-slice that prevents the decoder from referencing earlier slices and always occurs at the start of a new GOP [12]. SP and SI are special types of slices that are enabled through the Extended Profile of the H.264 standard. Our discussion is focused on the I/IDR-, P-, and B-slices as they are supported by nearly all H.264 profiles, and therefore, readily available in encoded videos.

H.264 offers abstraction of the bitstream in two layers: the Video Coding Layer (VCL) and the Network Abstraction Layer (NAL). The VCL refers to the actual compressed video that results from applying H.264 compression techniques (prediction, motion compensation, variable length coding, etc.). The NAL was introduced to support the packet-based nature of existing networks and dictates how the outside world (i.e., routers, NICs, network protocols) works with H.264-encoded video, without needing to know the details and specifications of the compressed video being transported [8].

When the H.264 syntax is mapped to frame sequences, each slice corresponds to a particular region of a frame. Fig. 2 shows an example frame that contains some missing slices due to packet loss. Slice boundaries are indicated by the solid blue lines, and it can be clearly seen that there are 8 slices. Note that decoders typically perform *error concealment* (EC) techniques when information is missing to hide

(a) The original frame.



(b) The received frame.

Figure 2: Effect of packet loss.

unwanted visual effects [12]. However, for the sake of illustration, the image shown is presented without error concealment. In this example, 32% of the packets are lost, but over half of the image is missing. Furthermore, the missing slices 2, 4, and 7 received most of their packets. However, loss of the first few packets in those slices rendered the entire slices undecodable. This illustrates the importance of header information present in each slice and NAL unit. Therefore, the proposed DRAS.264 exploits this important characteristic of H.264 video streaming to improve video reconstruction.

3. RELATED WORK

MAC layer retransmission for video streaming was first presented in [10], where retry limits were dynamically assigned to the layers of scalable-encoded video (MPEG-4 FGS[1]), but playout deadlines were ignored. More advanced methodologies were presented in [15, 16, 5]. These approaches perform offline computations to find optimal retry limits for various channel conditions and playout constraints; however, heuristic-based approaches were eventually use to meet real-time requirements. Extraction of packet importance is considered an integral part of these schemes and is typically done prior to the encoding process. In [11], packet deadlines were extended according to the importance of compressed video frames.

The general consensus that can be drawn from the aforementioned related work is that adaptive retransmission improves video and is necessary for real-time video streaming. However, none of these studies considered the current H.264 Advanced Video Coding (H.264/AVC) standard. Instead, MPEG-2 [5, 11] and custom codecs [15, 16] were analyzed, in part due to fine-grained characteristics that allow loss impacts to be determined on a per packet basis. Furthermore, video resolutions studied were no larger than CIF (352×288) at 30 fps, which pale in comparison to the demands of HD (1920×1080) video.

[1]MPEG-4 Fine-Granularity-Scalability

Although the related work discussed thus far has been on the MAC layer, there are also higher layer techniques to improve the streaming of the H.264 bitstream. The closest example of a higher level approach to our proposed DRAS.264 is the hybrid UDP/TCP scheme used in [18], which prioritizes H.264 bitstreams and then higher priority data are streamed over TCP while lower priority data are transmitted over UDP. Nevertheless, this method is orthogonal to the proposed DRAS.264 and thus can be used together.

The proposed DRAS.264 is designed for the current state-of-the art video compression standard, i.e., H.264. To the best of our knowledge, 802.11 retry limit adaptation for H.264 encoded HD video has not been explored. DRAS.264 exploits the Network Abstraction Layer (NAL) of H.264 to employ unequal error protection of packets containing slice headers, which are crucial to reconstruction of compressed video, as discussed in Sec. 2.2. In contrast to earlier schemes, DRAS.264 does not require video encoding information or network condition profiles beforehand. Instead, retry limits of packets are adapted dynamically according to network metrics representing backoff delay and contention. Furthermore, although previous retry adaptation methods have proven to be beneficial for wireless streaming of video, working with HD resolution, as opposed to CIF, proves to be much more challenging. This significantly expands the scope of research in the context of MAC layer retry adaptation.

4. DRAS.264

The proposed DRAS.264 scheme operates at the MAC layer to closely monitor 802.11 packet delays. Furthermore, real-time parsing of the H.264 bitstream is employed to allow for on-the-fly detection of packet importance. Based on this information, MAC retry limits are adjusted. When a packet is expected to exceed its playout deadline, it is simply discarded and not transmitted. Packets containing H.264 slice headers are given special treatment as they are allowed to be retransmitted either until received or the corresponding playout deadline is reached.

DRAS.264 works on MAC layer frames that contain H.264 encoded video encapsulated within Real-time Transport Protocol (RTP) packets. The packetization scheme adheres to the *Fragmentation Unit* (FU) structure specified in IETF RFC 6184 [17] with minor modifications to allow for Sequence Parameter Set (SPS) and Picture Parameter Set (PPS) to be grouped with slices. H.264 NAL units are fragmented into fixed-length RTP packets. Fig. 3 shows an example of the packetization scheme using RTP packets with maximum size of 1,450 bytes, where the header length is 12 bytes, leaving 1,438 bytes for the payload. As shown in the figure, not all packets will be 1,450 bytes in length due to the number of bytes within a slice having a non-integral multiple of 1,438 bytes. Furthermore, note that the FU structure is modified to allow for aggregation of parameter sets (i.e., SPS and PPS) with slices because of their small sizes.

The proposed DRAS.264 performs two main tasks: (1) *Retry Assignment* and (2) *Slice Protection*. Retry Assignment is responsible for modifying retry limits based on expected

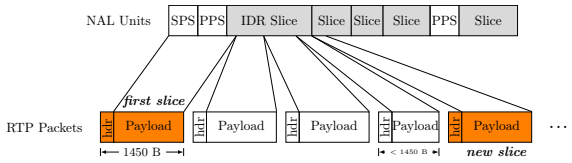


Figure 3: Modified IETF RFC 6184.

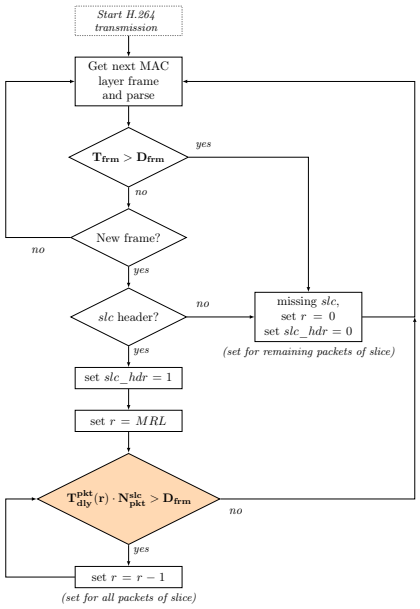


Figure 4: Flowchart for Retry Assignment.

packet delays and playout deadlines, and Slice Protection represents the decision making process on whether to drop packets containing slice headers or extend their retransmission opportunities.

Fig. 4 shows the Retry Assignment process of DRAS.264. Every MAC layer frame at the head of the transmission queue is processed to extract the encapsulated RTP packet. The RTP packet is then further parsed to extract the timestamp information. Note that all packets associated with a video frame share the same timestamp in the RTP header. Thus, a new video frame can be identified when a new timestamp is encountered. Finally, the RTP payload is parsed to determine its slice type and whether it contains a slice header.

The next step after parsing the MAC layer frame is to make a series of checks using the extracted information. The first critical check is to determine whether the projected delivery time of the current MAC frame (T_{frm}) will exceed the corresponding video frame playout deadline (D_{frm}), which is computed using the RTP timestamp (TS) obtained during the parsing step. The formula used to calculate D_{frm} is given as:

$$D_{frm} = T_{ini} + \lambda TS, \tag{1}$$

where T_{ini} is the initial startup delay and $\lambda = 1/fps$. For multi-slice video λ is divided by the number of slices per frame.

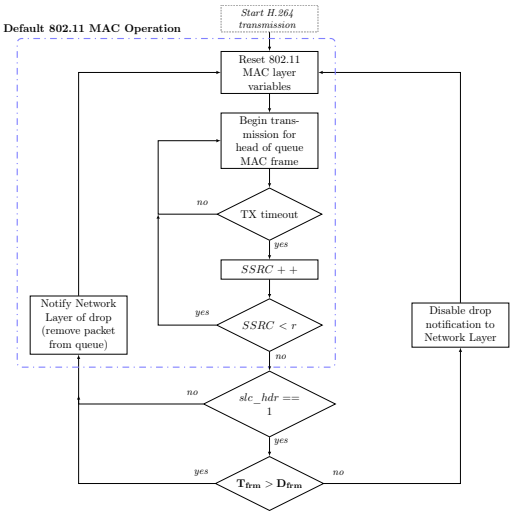


Figure 5: Flowchart for Slice Protection.

The Retry Assignment process takes place when a slice header is encountered. First, the slice header flag, *slc_hdr*, is set to indicate that the current MAC frame holds an H.264 slice header. Then, the Retry Assignment variable r is set to the *Maximum Retry Limit* (MRL), which is the maximum number of retransmissions permitted by the MAC layer. Next, r is used to index the array of predicted delays for a packet, $T^{pkt}_{dly}(r)$. The corresponding $T^{pkt}_{dly}(r)$ is multiplied by the number of packets contained in the slice, N^{slc}_{pkt}, where *slc* represents the slice type. Thus, N^{I}_{pkt}, N^{P}_{pkt}, and N^{B}_{pkt} represent the number of packets contained in I-slice, P-slice, and B-slice, respectively. Since these values are not known ahead of time, the maximum number of packets for each slice type is used. These are obtained by using the maximum values encountered per slice type up to the current transmission.

Typically, I-slices have the least amount of transmission time per packet because $N^{I}_{pkt} > N^{P}_{pkt} > N^{B}_{pkt}$. This conflicts with the level of importance exhibited by I-slices within an H.264 bitstream. Thus, the time normally allotted to B-slices (which has more transmission time per packet) is allocated to the packets of I-slices, and vice versa. This is done by using N^{B}_{pkt} for I-slices and N^{I}_{pkt} for B-slices when $T^{pkt}_{dly}(r) \cdot N^{slc}_{pkt}$ is compared to the frame deadline, D_{frm}, in the next step shown in Fig 4. For P-slices, N^{P}_{pkt} is used. This ensures the selection of retry limits is proportional to the impact those packets will have on the received video.

As $T^{pkt}_{dly}(r) \cdot N^{slc}_{pkt}$ is compared with D_{frm}, the value of r is continuously reduced by one until the projected cumulative packet delay $T^{pkt}_{dly}(r) \cdot N^{slc}_{pkt}$ is less than or equal to D_{frm}. When this occurs, r is no longer modified and retains its assigned value as the Retry Assignment process returns to parsing the next MAC layer frame until a new slice is encountered. The development of an analytical model for predicting $T^{pkt}_{dly}(r)$ is discussed in Sec. 4.1.

Fig. 5 shows the Slice Protection process of DRAS.264, which prevents link layer drops for selected MAC frames. As outlined in the figure, DRAS.264 applies the default 802.11

MAC protocol by monitoring the STA Short Retry Count (SSRC)[2], which is incremented after each transmission timeout (i.e. lost ACK frame). When $SSRC$ exceeds the assigned retry limit r set in Fig. 4, and if the *slc_hdr* flag is asserted indicating the current packet holds an H.264 slice header, the packet drop notification to the Network Layer is blocked under the condition that the projected delivery time T_{frm} does not exceed the frame deadline D_{frm}. When this occurs, the default MAC layer operation resets variables CW and BO, which effectively treats the current frame being transmitted as a completely new frame, despite exceeding the retry limit. Thus, DRAS.264 prevents all MAC frames containing slice headers from being dropped as long as they are within their corresponding playout deadlines. However, if T_{frm} exceeds D_{frm}, then the drop notification is permitted to inform the Network Layer.

4.1 Prediction of Packet Delays

This subsection presents an analytical model for estimating packet delay for the r^{th} retry attempt, $T_{dly}^{pkt}(r)$. As discussed in Sec. 2.1, a packet can be in one of the following four states: *success*, *collision*, *backoff*, and *deferred*. Each state represents a certain delay component that contributes to the overall packet delay from the perspective of a STA. We define the primary STA to be a STA that is streaming H.264 video with DRAS.264 enabled. Running averages of how long packets remain in each state are tallied for the purposes of prediction. It is important to note that packet delays for only *successfully received packets* can be obtained since delays for dropped packets are unknown. Three main metrics that affect the total delay a packet experiences from the time it is transmitted to the time it is successfully acknowledged are given below:

- n_c - number of collisions experienced by the primary STA sending the video;
- BO_r - the backoff slot chosen for the r^{th} retry attempt; and
- n_{tx} - number of transmissions by other STAs before the primary STA successfully receives an ACK.

These metrics are used to estimate the total delay a single packet at the head of the MAC layer transmission queue will experience, T_{dly}^{pkt}, which is given by the following equation:

$$T_{dly}^{pkt} = \sum_{r=0}^{n_c} T_{bkf}(r) + n_c T_{col} + n_{tx} T_{bsy} + T_{suc} + (n_c + n_{tx}) T_{DIFS} \quad (2)$$

The components of Eq. 2 cover the four main states of the 802.11 MAC layer operation, which are described in detail below:

- $T_{bkf}(r)$ represents the backoff delay associated with the r^{th} retry, and is defined as $T_{bkf}(r) = BO_r \times T_{slot}$. Note that $BO_r \in [0, CW_r]$, where $CW_r = a2^r - 1$. The constant a depends on the standard used. For example, 802.11a uses $a = 16$ and 802.11b uses $a = 32$.
- T_{col} and T_{bsy} both denote the time the medium is busy when a non-primary STA is transmitting. This can mean one of the two following possibilities: (1) the primary STA is in the deferred state and must wait for T_{bsy} or (2) a collision is taking place which wastes T_{col} amount of time. Both cases assume MAC layer frames used by all STAs in the ad-hoc network are of equal length. Based on this assumption, the following equation can be defined for T_{bsy}:

  $$T_{bsy} = T_{col} = T_{frm} + T_{SIFS} + T_{ACK},$$

  where T_{frm}, T_{SIFS}, and T_{ACK} are the MAC specific delays discussed in Sec. 2.1.
- T_{suc} is the time associated with successful transmission of a packet. Again, this is equivalent to T_{bsy} based on the assumption that all STAs transmit frames of equal length.
- T_{DIFS} is the MAC specific duration after each transmission for which the medium must be sensed idle before retransmitting.

Using the equality relation between T_{bsy} and T_{col}, Eq. 2 can be rearranged to the following equation:

$$T_{dly}^{pkt} = \underbrace{T_{bkf}(0) + T_{suc}}_{\text{Success}} + \overbrace{\sum_{r=1}^{n_c} T_{bkf}(r)}^{\text{Backoff}} + \underbrace{(n_c + n_{tx})(T_{bsy} + T_{DIFS})}_{\text{Collisions and Deferral}} \quad (3)$$

Eq. 3 ultimately represents the total delay a single packet at the head of the MAC layer queue experiences after n_c collisions and n_{tx} instances of deferrals.

In the default MAC layer protocol, a packet can only absorb a maximum of 6 retransmissions (7 total transmissions including the first attempt). Thus, each retry attempt has an average delay associated with it throughout the course of 802.11 network operation. This delay can be found by monitoring the *average backoff window size per attempt*, $E[BO_k]$, and the *average number of transmissions other STAs gain access to the medium*, $E[n_{tx_k}]$, for the k^{th} attempt. Furthermore, upon successful transmission of a packet, $n_c = r \leq R$, where R is the maximum number of retries (i.e., $R = 6$ by default). The estimated packet delay, $T_{dly}^{pkt}(r)$, can then be determined based on the number of retries using the following equation:

$$T_{dly}^{pkt}(r) = T_{suc} + \sum_{k=0}^{r} E[BO_k] T_{slot} + r(T_{col} + T_{DIFS}) + \sum_{k=0}^{r} E[n_{tx_k}](T_{bsy} + T_{DIFS}), \quad (4)$$

where T_{slot} is the duration of a time slot. Eq. 4 generates an array of predicted delays indexed by the retry attempt r. The running averages of backoff times $E[BO_k]$, and instances of deferral $E[n_{tx_k}]$ are summed for all attempts up to

[2]RTS/CTS is disabled for real-time video streaming

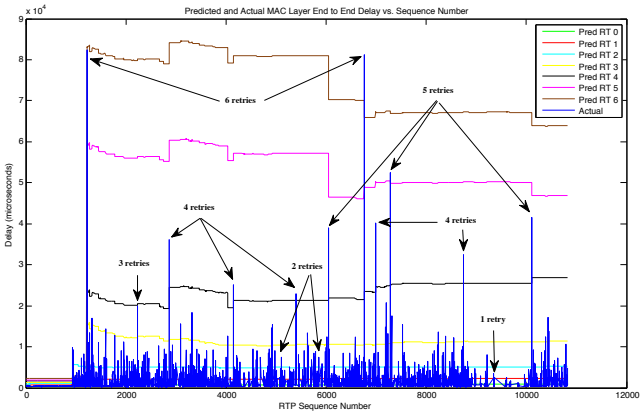


Figure 6: Actual MAC layer end-to-end delay and predicted delays for different retry attempts.

and including the retry attempt in question such that each prediction takes into account delays of previous attempts.

Fig. 6 shows how the predicted delays from Eq. 4 for different retry attempts compare to the actual MAC layer end-to-end delays (indicated in blue). The delays are for MAC layer frames transporting a 10 Mbps H.264 bitstream with three additional constant bitrate interference streams. As a reference, the number of retries needed for some of the actual end-to-end delays are pointed out on the figure. The predicted delays in Fig. 6 show that Eq. 4 tends to under-predict real delays experienced at the MAC layer. For example, in cases where four retries are needed, the corresponding predicted delays (indicated by solid black lines) are always less than the actual delay experienced. The only case where the delay is over-predicted is for five retries (indicated by solid pink line). Since over-predictions result in the selection of a retry limit too low for successful delivery, DRAS.264 employs some leniency in the retry assignment process. Thus, rather than using the hard limit resulting from the comparison $T_{dly}^{pkt}(r) \cdot N_{pkt}^{slc} > D_{frm}$ in Fig. 4, DRAS.264 assigns one higher value to r.

5. SIMULATION STUDY

Our simulation study was conducted using the *Open Evaluation Framework for Multimedia Over Networks* (OEFMON) developed at the Korea Advanced Institute of Science and Technology (KAIST) [9]. OEFMON is built upon a multimedia component *DirectShow* and a network simulator *QualNet* [13]. Together, they provide visualization of the underlying network details and on-the-fly display of sent and received videos. OEFMON requires the following inputs: A raw video file in YUV format, a QualNet scenario file, a QoS mapping parameter file, and a DirectShow graph. The three outputs generated are the received raw video file, a sender log, and a receiver log, which are used for offline analysis to compute PSNR, throughput, delay, and packet loss ratio among other metrics.

The simulation setup involves nodal arrangements of what would normally be found amongst neighboring apartments. Fig. 7 shows this type of arrangement with four pairs of streaming STAs. All distances between streaming pairs fall

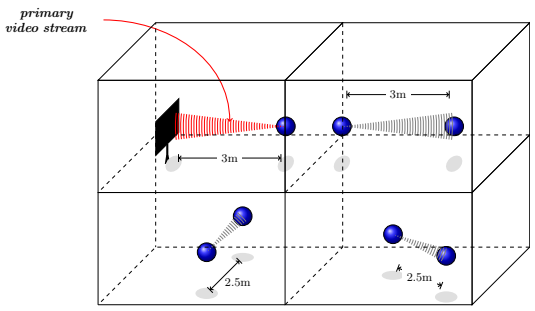


Figure 7: Experimental setup.

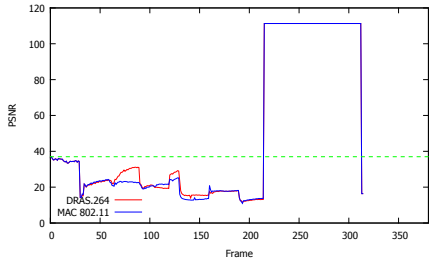
within 3 m, which is a reasonable viewing range in home networks. Note that this viewing range can vary; however, this does not impact performance. As long as transmit and receive pairs reside within the carrier sense range of each other, which can be as much as 100 m, there will be interference among video streams. The streaming pair shown in red represents the primary video stream for which DRAS.264 is implemented. All remaining streams simulate background traffic as a constant bitrate (CBR).

An 802.11a/g network with bandwidth 54 Mbps in QualNet 5.0.2 is used for simulation. Three test clips (*Sony Bravia*, *Heliboarding*, and *African Cats*) are used to represent the primary video streams. These clips are encoded using the *x264* open source H.264 encoder [3] Main Profile, Level 4.1 at 1080p @30 fps with an average bit rate of 10 Mbps. The length of test clips range from 315 frames to 372 frames. In addition, the streaming protocol used is RTP over UDP, and all CBR traffic is 10 Mbps resulting in a total aggregate bit rate of 40 Mbps to induce congestion in the network. The background streams start one second after the primary video stream is initiated and continue until the end of simulation when all the packets from the primary video stream have been sent. Furthermore, the current version of OEFMON is limited to single-slice encoding, thus an entire frame is encompassed in a single slice for the test video.

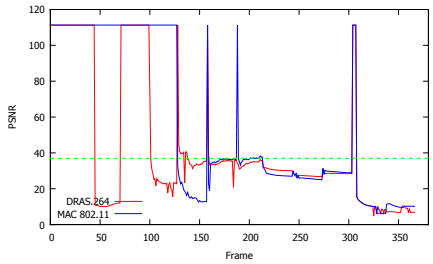
Note that these video clips have certain characteristics that need to be considered for proper streaming. For example, when streamed on a 54 Mbps wireless link without congestion, the initial startup delays are 58 ms, 85 ms, and 260 ms for the clips *Sony Bravia*, *Heliboarding*, and *African Cats*, respectively, to avoid missed playout deadlines. These requirements are a direct result of the bitrate variability (and thus jitter) of the encoded bit-streams, where frame size varies from 73.9 Kb to 3.3 Mb. The high startup time for African Cats is due to it's high frame size coefficient of variation (CoV) [14].

5.1 Results

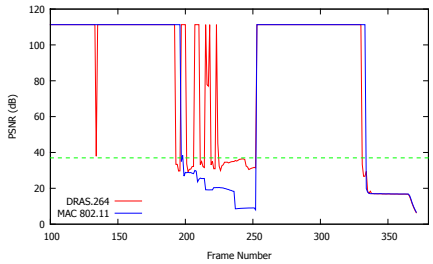
Fig. 8 shows PSNR values of the received videos in reference to the original undistorted video for DRAS.264 (red lines) and the default 802.11 MAC protocol (blue lines). The maximum PSNR value of 111 dB represents a perfectly received frame. A PSNR value of 37 dB is considered "excellent" quality [7] and is depicted by a dashed green line in each graph.



(a) *Sony Bravia.*



(b) *Heliboarding.*



(c) *African Cats.*

Figure 8: PSNR vs. Frame number.

It is important to note that there is no packet loss for the default 802.11 MAC operation. Instead, PSNR degradations are 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. Percentages of packets that miss their playout deadlines are 11%, 17% and 31% for *Sony Bravia*, *Heliboarding*, and *African Cats*, respectively. This phenomenon can be seen for the default 802.11 MAC operation in Fig. 8a (Frames 30-214), Fig. 8b (Frames 159-303, 308-366) and Fig. 8c (Frames 197-252, 334-371). Instances of sharp spikes (e.g., Fig. 8c Frame 253) and sharp dips (e.g., Fig. 8b Frame 128) are the results of reference slices (particularly I-slices) received in tact and missing information, respectively.

When DRAS.264 is applied to the primary video streams for the same network conditions, the percentages of total packets that miss their playout deadlines drop to 10%, 15%, and 18%, for Figs. 8a, 8b and 8c, respectively. This is due to two reasons: (1) reduced retry limits and (2) dropping expired packets. Thus, although packet loss occurs for DRAS.264, the total percentage of packet loss is less than half a percent for all three scenarios. This still allows DRAS.264 to have a lower *effective packet loss rate* than the default 802.11 MAC protocol. The resulting visual improvements are shown in Fig. 9 and can be explained in conjunction with PSNR results.

In Fig. 8a, the GOP containing Frames 30-59 has slightly lower PSNR values in comparison to the default 802.11 operation, which are caused by the loss of two packets that have exceeded their retry limits. Both losses have a minimal effect on the video, but allow additional time to deliver packets of the subsequent GOP containing Frames 60-89. These packets contain a sufficient number of MBs for the new IDR slice to have a better visual impact in the next GOP. In particular, this can be clearly seen in the part of the Killer Whale shown in Fig. 9a, which is preserved due to reduced error propagation for slices referencing the Whale object. This improves the PNSR values for the duration of the GOP. A similar phenomenon can be observed for the GOP containing Frames 128-157 in Fig. 8b. DRAS.264 is able to maintain higher PSNR values by delivering the initial IDR frame in tact and subsequent P-frames with minimal degradation as shown in Fig. 9b, which is the second P-frame of the GOP sequence.

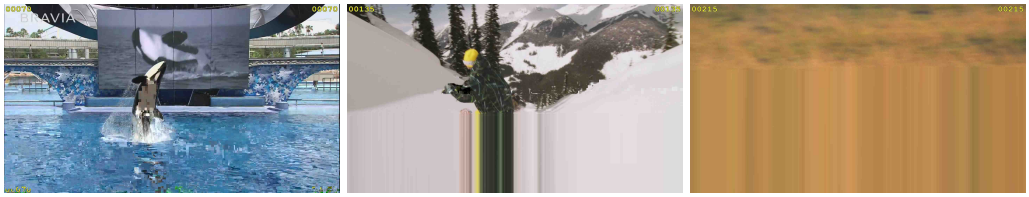
In Fig. 8c, a significant improvement in video quality is observed for multiple GOPs starting from Frame 197 due to a higher bitrate variability exhibited by *African Cats*. Similar to the other videos, some dips in PSNR values are observed prior to Frame 197. The corresponding visual impact of those dips are negligible considering their proximity to the green dashed line. However, for the sequence of frames up to and including 252, DRAS.264 preserves multiple I-frames (Frames 207, 215, and 223) yielding better video quality. The effect of this can be clearly seen by comparing the rightmost images in Fig. 9.

One drawback of the DRAS.264 algorithm is the over-prediction of per packet delays as discussed in Sec. 4.1 (see Fig. 6). When over-prediction occurs, more aggressive retry limits will be assigned to packets of I-slices leading to drops in PSNR, e.g., Frames 45-70 and Frames 101-127 in Fig. 8b. This can be mitigated by excluding I-slices from retry limit modification or developing a network congestion model that can determine when the DRAS.264 algorithm should be activated. We leave this as future work.

6. CONCLUSION AND FUTURE WORK

This paper presented DRAS.264, which is a MAC-based solution to improving real-time video quality for WHDVT. DRAS.264 employs a retry limit adaptation scheme tailored to H.264 content giving higher retry limits to packets containing reference slices, and even higher priority to packets holding H.264 slice headers by allowing indefinite retransmissions within frame deadlines. The added protection to slice headers allows a better chance for proper reconstruction of received video. Our results show trade-offs to the quality of lower impact frames in providing overall better video when DRAS.264 is compared to the static retry limit scheme of the default 802.11 MAC layer.

Several limitations to DRAS.264 will be addressed in future work. First, the simulation framework, OEFMON, works only on single-slice video. Therefore, OEFMON will be expanded to incorporate multiple slices. For multi-slice video, more packets per frame will be treated with the slice protection mechanism allowing DRAS.264 to have finer grain



(a) Default 802.11 MAC operation.

(b) DRAS.264.

Figure 9: Visual comparison of the default 802.11 MAC operation and DRAS.264. From left to right: *Sony Bravia* (Frame 70, B-frame), *Heliboarding* (Frame 135, P-frame), and *African Cats* (Frame 215, I-frame)

control over frame reconstruction. Second, DRAS.264 could benefit from a cross layer approach between the MAC and network layers to improve streaming. Specifically, monitoring the Network Layer queue to obtain the exact number of packets within a slice could help the accuracy in predicting retry limits. Furthermore, for cases where packets must be dropped due to expected deadline violations, access to the Network Layer queue can allow multiple packets to be discarded reducing delay.

Acknowledgements
This research 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).

7. REFERENCES

[1] http://www.intel.com/content/www/us/en/architecture-and-technology/intel-wireless-display.html.

[2] http://www.apple.com/airplay/.

[3] http://www.videolan.org/developers/x264.html.

[4] IEEE Std 802.11^{TM}-2007, IEEE Standard for Information Technology-Telecommunications and information exchange between systems-Local and metropolitan area networks-Specific requirements, June 2007.

[5] C.-M. Chen, C.-W. Lin, and Y.-C. Chen. Cross-Layer Packet Retry Limit Adaptation for Video Transport Over Wireless LANs. *IEEE Transactions on Circuits and Systems for Video Technology*, 20(11):1448–1461, 2010.

[6] 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 Computing*, 13(2):74–82, 2009.

[7] J. Gross, J. Klaue, H. Karl, and A. Wolisz. Cross-layer optimization of OFDM transmission systems for mpeg-4 video streaming. *Computer Communications*, 27(11):1044 – 1055, 2004. Applications and Services in Wireless Networks.

[8] M. Hannuksela and T. Wiegand. H.264/AVC in Wireless Environments. *Circuits and Systems*, 2003.

[9] C. Lee, M. Kim, S. Hyun, S. Lee, B. Lee, and K. Lee. OEFMON: An open evaluation framework for multimedia over networks. *IEEE Communications Magazine*, 49(9):153–161, 2011.

[10] 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 Transactions on Multimedia*, 6(2):278–290, 2004.

[11] M. Lu and P. Steenkiste. A time-based adaptive retry strategy for video streaming in 802.11 WLANs. *Wireless Communications and Mobile Computing*, 2007.

[12] I. E. Richardson. *The H.264 Advanced Video Compression Standard, Second Edition*. Wiley, Mar. 2010.

[13] Scalable Network Technologies, Inc. *QualNet 5.0.2 User's Guide*, 2010.

[14] P. Seeling and M. Reisslein. Video Transport Evaluation With H.264 Video Traces. *Communications Surveys & Tutorials, IEEE*, PP(99):1–24, 2011.

[15] M. van der Schaar, D. Turaga, and R. Wong. Classification-Based System For Cross-Layer Optimized Wireless Video Transmission. *IEEE Transactions on Multimedia*, 8(5):1082–1095, 2006.

[16] M. van der Schaar and D. S. Turaga. Cross-Layer Packetization and Retransmission Strategies for Delay-Sensitive Wireless Multimedia Transmission. *IEEE Transactions on Multimedia*, 9(1):185–197, 2007.

[17] Y.-K. Wang, R. Even, T. Kristensen, and R. Jesup. RTP Payload Format for H.264 Video. RFC 6184 (Proposed Standard), May 2011.

[18] J. Zhao, B. Lee, T.-W. Lee, C.-G. Kim, J.-K. Shin, and J. Cho. Flexible dual tcp/udp streaming for h.264 hd video over wlans. In *Proceedings of the 7th International Conference on Ubiquitous Information Management and Communication*, ICUIMC '13, pages 34:1–34:9, New York, NY, USA, 2013. ACM.