SEWS: A Channel-Aware Stall-Free WiFi Video Streaming Mechanism

Lixing Song, Aaron Striegel
University of Notre Dame, USA
{lsong2, striegel}@nd.edu

ABSTRACT

The rise of video streaming has placed significant demands on network infrastructure. These demands are most acutely felt in the wireless space where limited resources are available. Compounding the matter, most techniques for adapting to network dynamics have been developed with wired networks in mind thus making performance in congested wireless networks, especially WiFi, quite problematic. In this paper, we propose a novel cross-layer design to improve video bitrate selection by incorporating MAC layer information. We design a lightweight channel characterization method that can provide an accurate airtime estimation based on the observation of WiFi control packets. We then devise a bitrate adaptation algorithm that can judiciously avoid faulty bitrate increases whenever severe channel competition is detected. Through extensive lab experiments, we show that our proposed method can significantly reduce video stall rates by up to 30x (from 65% to 2%) compared to existing methods.

CCS CONCEPTS

• Information systems → Multimedia streaming; • Networks → Network measurement;

KEYWORDS

WiFi video streaming, cross-layer design, MAC layer, lightweight airtime estimation

ACM Reference Format:

Lixing Song, Aaron Striegel. 2018. SEWS: A Channel-Aware Stall-Free WiFi Video Streaming Mechanism. In *NOSSDAV'18: 28th ACM SIGMM Workshop on Network and Operating Systems Support for Digital Audio and Video, June 12–15, 2018, Amsterdam, Netherlands*. ACM, New York, NY, USA, 7 pages. <https://doi.org/10.1145/3210445.3210449>

1 INTRODUCTION

The Internet today has become an “eyeball economy” which is dominated by video traffic. According to Cisco [3], IP video traffic will be 82 % of all consumer Internet traffic by 2021. With recent advances in wireless network technologies (e.g., WiFi, cellular), video streaming via mobile devices has become the most common mechanism for distribution. Particularly, WiFi, with its high speed and

low cost, is the preferred choice for traffic-intensive services. Unfortunately, the feature of low cost as well as operating in unlicensed spectrum frequently force WiFi devices to deal with challenging and congested scenarios.

Furthermore, the growth in demand for WiFi devices shows no signs of abating. Traffic demand continues to rise, new devices emerge from the rise of the Internet of Things (IoT), and cellular operators are encroaching on existing WiFi spectrum with approaches such as LTE-U, LAA, and others. This becomes problematic for video streaming as the end result of these new demands is not only significant network congestion but more critically, networks that tend to experience significant dynamics. These new rapid dynamics that are present in WiFi networks tend to confuse many existing video adaptation schemes resulting in a higher likelihood of video stalling which in turn dramatically reduces the user Quality of Experience (QoE).

Fundamentally, the goal of adaptive video streaming scheme is to adjust the video bitrate based on the current network condition. An ideal algorithm requires an excellent prediction of the future throughput. However, WiFi networks, create havoc for achieving accurate throughput prediction. Notably, WiFi bandwidth is not only decided by the signal strength between sender and receiver but also by the usage of other nodes on the same channel. In a crowded WiFi environment, serious channel competition can make throughput vary significantly due to the various dynamics of the nodes present. This serious competition is best captured by a metric called *airtime*, which is the time duration of the channel being consumed by data transmissions. Unfortunately, most adaptation schemes use only the observed throughput from downloading the segments of the video stream and can miss out on the extent of competition dynamics on the wireless link¹. In such cases where the network is highly dynamic and subject to intense bursts of competition, operating in a conservative manner by choosing the lowest video bitrate can avoid stalling at the cost of a low video resolution/quality.

Thus, the premise of the paper is relatively straightforward: *Can we design an adaptation algorithm that recognizes overwhelming congestion to operate conservatively while being aggressive during amenable times to maximize video quality?* In this paper, we propose the **Stall-free WiFi Streaming mechanism SEWS**. The proposed work adopts a cross-layer design that exploits MAC layer information to facilitate video bitrate adaptation. We design a novel lightweight channel traffic characterization method that enables per-link channel airtime estimation. The characterization method is straightforward in that it only requires capturing one type of control packets – the *Block Acknowledgement* – to achieve a high-accuracy

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than ACM must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

NOSSDAV'18, June 12–15, 2018, Amsterdam, Netherlands

© 2018 Association for Computing Machinery.

ACM ISBN 978-1-4503-5772-2/18/06...\$15.00

<https://doi.org/10.1145/3210445.3210449>

¹In fairness to most schemes, many operate at upper layers while the purpose of our paper is to show how that can be particularly deleterious for WiFi.

airtime measurement. Based on fine-grained airtime information, we further devise an adaptation algorithm that can judiciously avoid bitrate over-selection for certain channel condition. The proposed algorithm can significantly reduce video stalling while maintaining acceptable video quality. Overall, the main contributions of this paper can be summarized as follows:

- **Lightweight Channel Characterization.** To enable channel awareness for video streaming, we design a novel lightweight channel traffic characterization scheme that only requires observing one type of control packet – *Block Acknowledgement*. We find that the time interval of these packets allows an accurate inference of channel airtime. In addition, the address information embedded in the packets enables a fine-grained characterization in per-link resolution. We conduct lab experiments to validate the estimation accuracy of the designed method.
- **Stall-Free Adaptation Algorithm.** By exploiting the airtime information obtained from the method above, we propose a cross-layer WiFi video streaming algorithm that helps prevent faulty bitrate selection. The algorithm can effectively avoid bitrate over-selection when a certain level of channel competition is detected.
- **Implementation&Experimental Evaluation.** To evaluate the performance, we modify an open-source DASH client (*GPAC* [5]) to pass the channel characterization information (parsed from WiFi adapter) to facilitate video bitrate selection. By comparing the lab experiment result with several existing flagship works [5–8], we show that our proposed method can achieve up to a 30x stall rate reduction (from 65% to 2%) while maintaining an acceptable video quality.

2 BACKGROUND & MOTIVATION

To motivate our work, we present an overview of DASH and conduct a preliminary experimental study to demonstrate the extent of the problem video stalling on WiFi networks.

DASH Overview. DASH (Dynamic Adaptive Streaming over HTTP) is an adaptive bitrate streaming technique. It broadly refers to a class of protocols that provide media content streaming via an HTTP web server, including Apple's HTTP Live Streaming (HLS), MPEG-DASH, Microsoft Smooth Streaming, etc. In general, DASH breaks media content into a sequence of small segments and each segment contains a short interval (typically 1 to 10 seconds) of the video. By pre-encoding and storing the video in a number of discrete bitrate levels, a client can adjust the video quality for each segment by selecting different bitrate copies.

Bitrate selection requires a throughput prediction for the next segment download. Under-selection sacrifices video quality. While over-selection may result in video stalling (or freezing). Many existing adaptation algorithms [5, 7, 8] leverage the throughput history to assist bitrate selection. Others [6, 13] try to exploit the video buffer as a reference to adjust bitrate. Most of the prior works primarily target wired networks. However, due to the different channel properties of wireless links, video streaming can experience different bandwidth variations.

WiFi Video Streaming Experimental Study. We conduct a controlled lab experiment to demonstrate video streaming performance under busy WiFi. The experimental setting is as follows: we set up

a client to stream video content from a local server via a WiFi link (i.e., 802.11ac on 5GHz). On the same WiFi channel, we generate competing traffic (i.e., long-lived TCP flows) from other WiFi clients. In this experiment, while streaming 100-second video content, we launch two competing clients at the 20 and 40 seconds respectively. Then, competing traffic is stopped at the 80 seconds. We defer other detailed settings to the later section at Sec. 4. We show the stall rate (the percent of the time when video buffer is drained out) for several popular bitrate adaptation algorithms[5–7]. For reference, we also tested fixed bitrate approaches, i.e., the minimum (3Mbps) and maximum (52Mbps) used in Sec. 4.

Table 1: Stall rate for different algorithms

Algo.	min Bitrate	Latest Thrpt.	Thrpt. mean*	Buffer- based	max Bitrate
Stall Rate(%)	2.12	51.85	45.50	47.76	73.07

*harmonic mean.

Result: Table 1 lists the stall rate for the different algorithms. First, we see that the bandwidth is sufficient to support the minimum bitrate with a negligible 2% stall rate. The maximum bitrate causes significant stalling (73%) as expected. The three existing algorithms all experience unacceptably high stall rate (around 50%). Specifically, the highly variant throughput on WiFi makes the latest throughput approach [5] hard to predict. Although smoothed throughput [7] can help stabilize the bitrate, the stale reaction to bandwidth variation can still lead to severe stalling. The dynamic throughput leads to fast-changing buffer level which hampers the buffer-based approach [6].

Figure 1: Time series plot of the latest throughput-based approach. The three vertical lines are the starts of the first and second competing client, and the end of all competing traffic respectively. The shadow box at (b) indicates video stalling.

Problem Diagnosis. As mentioned earlier, the throughput on the WiFi link can be largely captured by the airtime on the channel. Thus, we further look into the airtime behavior during the streaming experiment described above. In Fig 1, we plot the time series of the latest throughput-based adaptation algorithm. The subplots are (a) the segment throughput/bitrate selection (b) the instantaneous video buffer (c) the WiFi channel airtime. The airtime is estimated by our proposed method which will be introduced in the next section (Sec 3). Particularly, we break down the airtime consumption for competing traffic and video traffic. As shown, without competing traffic (i.e., 0-20 s and 80-100 s), the video can be smoothly played on the highest bitrate. When competing with other clients (from 20 s

onwards), the WiFi channel starts to become crowded and the video client starts to experience a significant throughput drop. However, what causes the stalling is not the low throughput since we knew that the minimum bitrate is well-supported (Table 1). Instead, the root cause is due to the faulty bitrate selection when throughput fluctuates. The transient high throughput under channel congestion consistently misleads the video client into over-selecting, which results in video stalling. In addition, frequently switching the bitrate further exacerbates the variation of throughput.

Implication: From the experimental observation above, we can see that predicting the highly-transient bandwidth under congested WiFi is impractical. Instead, under this circumstance, conservatively staying at a fixed bitrate (e.g., minimum) can potentially avoid over-selection and aggressive switching. Therefore, we argue that in order to avoid video stalling, a video client needs to *infer channel competition on WiFi and act accordingly*. In the next section, we will introduce the proposed channel-aware cross-layer bitrate adaptation mechanism.

3 SYSTEM DESIGN

As one of the primary contributions of this paper, we will first introduce our novel approach to calculate a fine-grained channel airtime characterization (at Sec 3.1). Based on the channel information, we further devise the Stall-free WiFi Streaming algorithm *SEWS* (at Sec 3.2).

3.1 Fine-Grained Channel Airtime Characterization

On WiFi, the packets transmitted on a particular channel can be heard by nearby devices. This property enables channel characterization on WiFi in a passive manner. Since channel airtime is a crucial indicator of the channel traffic condition, our goal is to find a non-intrusive (i.e., passive) approach to obtain airtime estimation. Existing solutions usually rely on the spectrum sensing function available on WiFi access point (AP) [10]. However, since DASH streaming is a client-centered approach, we expect a client-side solution. A brute-force approach is to capture most if not all traffic transmitting on the channel. Unfortunately, regardless of the energy consumption, capturing all traffic is often impractical, due to many technical challenges, such as the bandwidth capacity difference of devices², beamforming, etc. Fortunately, we find that one type of control packet – Block Ack – can innately reveal the airtime usage on the WiFi channel.

Figure 2: Illustration of data transmission under frame aggregation.

A Primer on Block Ack. Block Ack was introduced in 802.11e as part of the *frame aggregation* transmission scheme. Frame aggregation can assemble multiple data units to transmit as one aggregated frame called aggregate MAC protocol data unit (*A-MPDU*).

²A low capacity device (e.g., low bandwidth, low supported rate) cannot capture traffic transmitted from high-capacity devices.

In tandem with frame aggregation, a Block Ack is used to confirm the receipt of an A-MPDU which can contain multiple MPDUs (i.e., packets³) as shown in Fig 2. As frame aggregation has been adopted as the default manner of data transmission on modern WiFi (802.11n/ac), data traffic will always invoke exchanges of Block Acks. Therefore, the observation of Block Ack embodies rich information regarding data traffic [12].

Calculating Airtime. Block Ack targets for coordinating data transmission. The occurrence of Block Ack must be in tandem with data transmission. Intuitively, we can infer the time consumption of data transmission by exploiting the time interval between consecutive Block Ack pairs. Therefore, given the Block Acks $BA_i, i = 1, 2, \dots$ captured in a time window ω , we can compute the airtime as the percentage of the time when the channel is busy.

Thus, $Airtime = \frac{\sum \omega G_{BA}^i}{\omega}$, where $G_{BA}^i = t(BA_{i+1}) - t(BA_i)$ is the time gap between a pair of consecutive Block Acks on a channel. $t(BA_i)$ is timestamp of the i -th captured Block Ack. Noted that these consecutive Block Acks can originate from different transmitters and be sent to different receivers.

Per-link Resolution: With the transmission address (TA) and receiver address (RA) labeled in Block Ack, the airtime estimation can be broken into a per-link scale. A link is indicated by a tuple of TA and RA. For example, between a mobile client and its associated AP, the traffic can occur on either the uplink (from client to AP) or the downlink (from AP to client). This characterization enables in-depth perception towards the airtime consumption on the WiFi channel.

3.2 SEWS: A Stall-Free WiFi Streaming Adaptation Algorithm

According to our discussion earlier, the bitrate over-selection caused by transient throughput spikes during channel competition is the culprit of video stalling. The goal of our algorithm is to avoid faulty bitrate increases under the congested WiFi channel. The proposed algorithm will use the latest throughput as the baseline. The primary task is to identify the problematic case when the throughput increase is a false signal for bitrate increases. By analyzing the airtime on WiFi, we find that the erroneous bitrate increasing usually occurs under *two conditions*: 1) *the channel is fully occupied and 2) the video streaming link is not the dominant link (i.e., the highest airtime link)*. Namely, if the two conditions occur simultaneously, a video client should *not increase bitrate* regardless of the throughput.

When the channel is not fully consumed, the throughput-based approach normally has no issue. When the channel is fully occupied, the throughput on video link can still increase to suppress the other traffic *if and only if* the video link is the most aggressive link that dominates the airtime consumption. Therefore, when two conditions are triggered, the video client should be precluded from bitrate increasing.

Overall, the algorithm can be described as follows in Algorithm 1. For each video segment, the algorithm will take the following information as input: the current video bitrate *cur_rate*, the measured throughput for the previous segment *Thrpt*, the airtime $\mathcal{A}$ measured during the previous segment playback time. We estimate

³Packet and MPDU are used interchangeably for the rest of the paper.

airtime $\mathcal{A}_i$ ($i = 1, \dots, N$) for each active link on the channel. Particularly, we denote the video link airtime as $\mathcal{A}_v$. In addition, we need the set of bitrate options $\mathcal{R}$ for all levels of rate $j = 1, \dots, M$. At last, we define a fully-occupied channel when overall airtime is beyond α .

Algorithm 1: Bitrate Adaptation Algorithm

```

Data:  $cur\_rate, Thrpt, \mathcal{A}, \mathcal{R}, \alpha$ 
1 not_increase = false
2 for  $i = 1$  to  $N$  do
3   if  $\mathcal{A}_i > \mathcal{A}_v$  and  $\sum \mathcal{A} > \alpha$  then
4     not_increase = true
5   break
6  $new\_rate = 0$ 
7 for  $j = 1$  to  $M$  do
8   if  $Thrpt > \mathcal{R}_j$  then
9      $new\_rate = \mathcal{R}_j$ 
10 if  $new\_rate > cur\_rate$  and not_increase then
11   return  $cur\_rate$ 
12 else
13   return  $new\_rate$ 

```

As described in Algorithm 1, the *not_increase* signal is used to control whether bitrate increasing is allowed based on the two criterion discussed earlier. In the following section, we will conduct extensive lab experiments to evaluate the performance of the proposed scheme under various realistic network settings.

4 PERFORMANCE EVALUATION

Figure 3: Implementation software work flow.

System Implementation: To conduct our experiments, we implemented the system on a Commercial off-the-shelf (COTS) device. The overall system work flow is shown in Fig 3. To capture the Block Acks for airtime estimation, the WiFi adapter needs to eavesdrop on the channel while maintaining communication. This requires a firmware modification which is impractical on commodity devices. Therefore, for evaluation purposes, we employ an extra USB WiFi adapter in monitor mode to conduct airtime characterization. This extra adapter is plugged into the video client laptop. The airtime information is fed to the video client through a messaging library *zmq*⁴.

Experiment Settings: We set up a DASH server via Apache on a laptop (HP ProBook 4730s). The server was connected to a controller (another HP laptop) which was used to configure the network environment (e.g., round trip time—RTT). WiFi clients connected to the controller via WiFi AP(s). We used TP-Link Archer C7 V2 802.11n/ac as the AP(s), and HP 250 G6 notebook (equipped with Intel Wireless 3168 802.11ac chipset) as the clients. For the video client, we used an Edimax EW-7822UAC USB adapter for airtime characterization purpose. The experiments were primarily run on 802.11ac 5G channel (channel no. 149). We set up two identical APs on the same channel to create competition. The WiFi link can achieve up to 150Mbps throughput with 20ms RTT. For the video streaming setting, the DASH server could provide 7 different

⁴zeromq.org

bitrates, including 3, 7, 17, 27, 32, 37, 42, 52Mbps. The segment length was set to 1 second and the client buffer length was 2000 ms as the low latency setting. We set $\alpha = 0.86$ based on empirical observations. Due to space constraints, we will not present the detailed results of searching the setting of α .

4.1 Design Validation

Figure 4: (a) The estimated airtime and the ground truth w.r.t. flow size and client number. (b) CDF of the airtime estimation error.

Airtime Evaluation: We start with an experimental validation of the airtime characterization method. By adopting the experiment setting above, we generated different sizes of TCP flows to consume different amounts of airtime on the WiFi channel. We varied the flow size from 20KB to 160MB. For each flow size, we kept repeating the flow in a back-to-back manner. Each flow ran for 100+ seconds. We employed multiple clients to generate the traffic simultaneously. The ground truth airtime can be polled from the WiFi AP driver (i.e., *ath9k*) via *iw* tool. We set a WiFi client in the monitor mode to capture the Block Acks and estimated the airtime. We set the window size $\omega = 1$ s as the video segment length in this case.

Fig 4a plots the estimated airtime as well as the ground truth under the different flow sizes and client numbers. We see that, regardless of the number of clients, the estimated airtime closely follows the ground truth across different flow sizes. However, we notice a slight underestimation from our method. It is because our approach only measures data packet consumption which leaves the control and management packets ignored. Fortunately, since the data transmission is the primary cause of channel usage, this underestimation is negligible. By calculating the estimation error (subtracting the estimated result from the ground truth), we see that the proposed method can restrict the estimation error within $\pm 10\%$ for over 80% estimations (as shown in Fig 4b).

4.2 System Micro-Benchmark Test

We continue to conduct system-level micro-benchmark tests and comparison with the existing flagship works. We selected five contrast DASH adaptive algorithms: *BBA*[6], *FESTIVE*[7], *PANDA*[8], *GPAC*[5] and *minRate*. *BBA* is a buffer-based approach that maps the instant buffer level to certain video bitrate adjustment. We adopted the *BBA-0* implementation proposed in [6]. *FESTIVE* uses the harmonic mean of throughput history to guide bitrate selection. We set the smooth window to 5 seconds. *PANDA* behaves like TCP AIMD (Additive Increase Multiplicative Decrease): it keeps increasing bitrate with an additive term; once throughput decreases, it reduces the bitrate with a multiplicative term. The parameters are tuned based on [8]. *GPAC* leverages the latest throughput to select

Figure 5: Performance comparison under heavy-flow competing traffic.

Figure 6: Performance comparison under on-off light/medium-flow competing traffic.

video bitrate. For reference purpose, we also ran the fixed bitrate by using the minimum bitrate.

Evaluation Metrics: Three metrics are employed to evaluate the performance, including *stall rate*, *smoothness* and *video bitrate*. *Stall rate* is the percent of the time when video buffer is drained out. *Video bitrate* computes the average of bitrate selected over entire video. *Smoothness* measures the intensity of bitrate switching which can be computed as the average absolute video bitrate change per segment. A good adaptive algorithm should try to stabilize the bitrate and yield a low smoothness value.

Heavy Flow Stress Test

We first evaluate the video adaptation algorithms under heavy-flow competition. The experiment started with streaming a 100-second on a clear WiFi channel. At 20 second, the competing client launched a long-lived TCP flow by fetching a big file from the server. The competing traffic lasts for 60 seconds and stops. We varied the number competing clients from 1 to 5. For each adaptation algorithm under a certain client number, we repeated the experiment over five times.

Results Analysis: Fig 6 plots the performance comparison of all the algorithms. In the figures, the boxes are the means and the bars denote standard deviations. We selectively plot the results from 1, 3, and 5 competing clients. First, as one of the most critical QoE metrics, we see that the stall rates in Fig 6a are unacceptably high (> 0.4) for all approaches except for the *minRate* and the proposed method. The stall rate worsens when the client number increases. Overall, our method can maintain the stall rate within 0.2 even under severe competition with 5 competing clients. Compared to the second-best solution (*BBA*), it achieves up to 20x (0.41 v.s. 0.02)

performance gain under the 1 competing client. *FESTIVE* ranks right behind *BBA*. Its dull reaction to bandwidth variation, due to the smoothed throughput measurement, results in high stall rate. *PANDA* aggressively raises bitrate to probe bandwidth which leads to the highest under this congested network. *GPAC* is sensitive to throughput fluctuation which makes it utterly vulnerable to faulty bitrate over-selection.

We continue to compare video bitrate and smoothness. From Fig 6b, we see that, with the similar low stall rate compared to the *minRate*, our proposed method can offer 7x video quality gain. Although the other adaptation algorithms provide better bitrate, the cost of stalling is undesirable. Because video stalling is more detrimental than low video bitrate from the QoE standpoint. The high video quality is always running the risk of bad stall rate. For instance, *PANDA* has the highest overall bitrate due to its aggressive probing but it provides barely watchable video with up to 0.78 stall rate. With more competing clients, video quality decreases because of the less available airtime on the channel. Note that, the bitrate of our method is relatively consistent across different numbers of competing clients. It is because once a certain level of competition is detected, our method usually prefers to staying at a low rate. This behavior also brings good video smoothness. As we can see in Fig 6c, the proposed method achieves the best smoothness expect for the *minRate*. Overall, as expected, the approaches (i.e., *FESTIVE*, *PANDA*) that adopts smoothed measurement yield better smoothness than the other approaches (i.e., *BBA*, *GPAC*).

On-Off Light/Medium Flow Test

In normal traffic environment, light/medium-flow traffic can be more prevalent than heavy-flow traffic. Therefore, we conducted

a further evaluation under light/medium flows in the WiFi environment. Similarly, in this experiment, we started with streaming a 120-second video on a clear WiFi channel. Meanwhile, the competing client(s) generated traffic lasting for 20 seconds and then slept for 20 seconds recursively. For each competing client, the competing traffic was generated by launching a TCP flow with the flow size randomly selected from three values (i.e., 2MB, 8M, 160MB). Again, we varied the competing client number from 1 and 5. For each client number and adaptation algorithm, we repeated the experiment over five times.

Results Analysis: As the result shown in Fig 7, the overall stall rate is much milder than the heavy-flow traffic scenario (Fig6a). The general stall rate ranking is unsurprisingly similar to the previous scenario. However, in term of video bitrate (Fig 7b), we see that our proposed method can achieve comparable video quality as other methods while maintaining extremely low stall rate. For example, under 3 competing clients, our proposed method can yield 73% (35Mbps v.s. 48Mbps) out of the highest bitrate solution *PANDA*. But we can achieve four times of stall rate reduction from 0.47 to 0.11. Namely, the significant reduction in stall rate only costs a small amount of sacrifice on video bitrate. In terms of smoothness, the observation in Fig 7c reveals similar pattern as the previous case. It is noteworthy that when the number of clients increases, the smoothness becomes worse in this case. It is because the on-off pattern can force more bitrate switching. In summary, the experimental evaluation proves that the proposed method can effectively reduce the stall rate while keeps acceptable bitrate in various WiFi traffic conditions.

5 RELATED WORK

Cross-Layer Video Streaming: Video bitrate adaptation algorithm has been a well-searched area in the past decades. In addition to the approaches solely working on the upper layers [5–8, 13], there are many works that attempt to tackle this problem in a cross-layer manner. For example, [11] proposed a framework to leverage different protocol layers to improve the end-to-end real-time video streaming performance. Particularly, under the mobile network, many works focus on extracting link / physical layer information to improve video streaming performance. For example, a recent work [14] proposed a novel approach to overcome the bandwidth underutilization of video streaming on LTE network by manipulating the physical layer information. Similarly, our paper aims to mitigate the video stalling problem on WiFi networks. In contrast to the other cross-layer works with expensive/complicated implementation cost, our work is easily adapted to work on commodity device due to the lightweight design.

Channel Characterization: As another primary contribution of this paper—channel characterization, the existing work mostly relies on the AP side [1, 10]. The passive client monitor methods are usually designed for a particular purpose. For example, [2] manipulates smartphones to monitor WiFi traffic in order to infer human activities. For general purpose traffic characterization, many prior works [4, 9] focus on merging traffic or inferring unobserved traffic. However, they normally require capture most if not all data traffic. Regardless of the intensive power/computation consumption of those methods, they could suffer from low accuracy due to the high

capture loss under the modern WiFi environment today. In contrast, our method merely requires one type of control packet to conduct comprehensive traffic characterization which is more efficient and accurate than the previous approaches.

6 CONCLUSION

In this paper, we proposed a cross-layer video bitrate adaptation mechanism targeted at WiFi networks. The designed method focused on reducing video stalling under congested WiFi environments. By exploiting a lightweight passive WiFi channel characterization, we showed that our proposed method can accurately measure the instantaneous airtime consumption on a WiFi channel. We further designed a bitrate adaptation algorithm (*SEWS*) that judiciously avoids faulty bitrate selection during severe channel competition and showed through lab experiments how our solution yielded a nearly 30x reduction in stall rate. Future work includes exploring advanced levels of MIMO for WiFi and newly emerging WiFi standards.

REFERENCES

- [1] S. Biswas, J. Bicket, E. Wong, R. Musaloiu-E, A. Bhartia, and D. Aguayo, "Large-scale measurements of wireless network behavior," *SIGCOMM Comput. Commun. Rev.*, vol. 45, no. 4, pp. 153–165, Aug. 2015.
- [2] Y. Chon, S. Kim, S. Lee, D. Kim, Y. Kim, and H. Cha, "Sensing wifi packets in the air: Practicality and implications in urban mobility monitoring," ser. UbiComp '14. New York, NY, USA: ACM, 2014, pp. 189–200.
- [3] Cisco. (2016) Cisco visual networking index: Forecast and methodology, 2016–2021. [Online]. Available: "https://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/complete-white-paper-c11-481360.html"
- [4] D. Da Hora, K. Van Doorselaer, K. Van Oost, R. Teixeira, and C. Diot, "Passive wi-fi link capacity estimation on commodity access points," in *Traffic Monitoring and Analysis Workshop (TMA) 2016*, 2016.
- [5] ENST. GPAC. [Online]. Available: "https://github.com/gpac/gpac"
- [6] T.-Y. Huang, R. Johari, N. McKeown, M. Trunnell, and M. Watson, "A buffer-based approach to rate adaptation: Evidence from a large video streaming service," ser. SIGCOMM '14, New York, NY, USA, 2014.
- [7] J. Jiang, V. Sekar, and H. Zhang, "Improving fairness, efficiency, and stability in http-based adaptive video streaming with festive," *IEEE/ACM Transactions on Networking*, vol. 22, no. 1, pp. 326–340, Feb 2014.
- [8] Z. Li, X. Zhu, J. Gahm, R. Pan, H. Hu, A. C. Begen, and D. Oran, "Probe and adapt: Rate adaptation for http video streaming at scale," *IEEE Journal on Selected Areas in Communications*, 2014.
- [9] A. Mahanti, C. Williamson, and M. Arlitt, "Remote analysis of a distributed wlan using passive wireless-side measurement," *Perform. Eval.*, vol. 64, 2007.
- [10] A. Patro, S. Govindan, and S. Banerjee, "Observing Home Wireless Experience Through WiFi APs," ser. MobiCom '13. New York, NY, USA: ACM, 2013, pp. 339–350.
- [11] E. Setton, T. Yoo, X. Zhu, A. Goldsmith, and B. Girod, "Cross-layer design of ad hoc networks for real-time video streaming," *IEEE Wireless Communications*, Aug 2005.
- [12] L. Song and A. Striegel, "Leveraging frame aggregation for estimating wifi available bandwidth," in *2017 14th Annual IEEE International Conference on Sensing, Communication, and Networking (SECON)*, June 2017, pp. 1–9.
- [13] K. Spiteri, R. Ugaonkar, and R. K. Sitaraman, "Bola: Near-optimal bitrate adaptation for online videos," in *IEEE INFOCOM 2016*, April 2016.
- [14] X. Xie, X. Zhang, S. Kumar, and L. E. Li, "pistream: Physical layer informed adaptive video streaming over lte," ser. MobiCom '15. New York, NY, USA: ACM, 2015, pp. 413–425.