A Frame-Aggregation-Based Approach for Link Congestion Prediction in WiFi Video Streaming

Shangyue Zhu, Alamin Mohammed, Aaron Striegel
Department of Computer Science and Engineering
University of Notre Dame, U.S.A
{szhu7,amohamm2,striegel}@nd.edu

***Abstract*—Video streaming using WiFi networks poses the challenge of variable network performance when multiple clients are present. Hence, it is important to continuously monitor and predict the network changes in order to ensure a higher user quality of experience (QoE) for video streaming. Existing approaches that aim to detect such network changes have several disadvantages. For example, active probing approaches are expensive so that generate more additional traffic flow during the testing. To overcome its shortcomings, we propose a passive, lightweight approach, *CP-DASH*, whereby queuing effects present in frame aggregation are leveraged to predict link congestion in the WiFi network. This approach allows the early detection which can be used to adapt our video appropriately. We conduct experiments simulating a WiFi network with multiple clients and compare CP-DASH with five contemporary rate selection mechanisms. We found that our proposed method significantly reduces the switch rates and stall rates from 22% to 5% and from 38% to 25% compared with an existing throughput-based algorithm, respectively.**

***Index Terms*—Video Streaming, Frame Aggregation, link Congestion**

I. Introduction

The proliferation of devices using WiFi has led to increased network traffic bringing increased network congestion and significant variations in network performance. With this increasing number of devices competing for limited network resources, significant strains emerge with the the most dominant traffic type, namely video. Per a recent forecast [1], IP video traffic represented 75% of all traffic in 2017 and is slated to continue to grow to over 82% of all IP traffic (both business and consumer) by 2022. An on-going and critical challenge is how to handle one of the most temperamental video traffic, the need for uninterrupted, high-quality video streaming in the presence of rapidly shifting, overloaded network dynamics.

Methods for adaptive video streaming generally aim to set the video bit-rate appropriate for the current network conditions. A very conservative approach [2] can be used wherein the lowest bit-rate is chosen but this approach has the drawback of providing a low bit-rate video even during the times when the network is less congested. There are also approaches that aim to detect the throughput, for example, via chunk download performance [3]. Unfortunately, while measuring intermediate download speed can prove useful, such information can often be quite late (e.g. one misses the rapidly-occurring network changes), and/or noisy (e.g. captured rate changes requires smoothing or filtering). These negative effects will potential degrade the video quality even cause economic losses.

Monitoring the network performance, especially when the network is unstable, can be nontrivial. Active probing is one approach for establishing connections and observing the status of the channel [4]. Active probing is considered to be an effective method for monitoring the network status and uses conventional network measurements, such as available bandwidth and link congestion, potentially without the delayed reporting associated with block download speeds. However, when channel resources are already limited, excessive (active) probing can cause more network congestion, making a bad problem even worse [5].

Given the above drawbacks of existing approaches, in this paper, we propose a novel approach that detects link congestion in an accurate, timely but passive manner to deliver a better rate adaptation mechanism for adaptive video streaming. We leverage the aggregate MAC protocol data unit (*A-MPDU*) information provided by *Block Acknowledgement* aggregate control packets to allow a WiFi device passively predict link congestion. We take advantage of queuing effects present in frame aggregation to get an early warning signal which provides an accurate estimation of network performance that we in turn use to adapt our video rate appropriately.

In this paper, we propose an algorithm, *Congestion Prediction-DASH (CP-DASH)*, building on the DASH technique of Fuzzy DASH (F-DASH) [6] which also uses a buffer-based method to reduce video stalling and to provide high-quality streaming. The main contributions of this paper are as follows:

- *A passive, lightweight approach for predicting link congestion:* To detect link congestion for video streaming, we designed a novel method that only captures control packets or *Block Acknowledgement* (Block Acks). From extensive performance evaluations, we show that the network-condition information extracted from Block Acks can effectively be used for link-congestion prediction.
- *CP-DASH algorithm:* We combine the above link-congestion prediction with a buffer-based algorithm for selecting an appropriate bit-rate for the video. Our algorithm, *CP-DASH*, is designed to prevent stalling and

to maximize video quality. Furthermore, by being buffer-based, our approach can effectively prevent bit-rate over-selection when the buffer space is almost depleted.

- *Implementation and Evaluation (presented in Section IV):* We conduct experiments by simulating a WiFi network with multiple clients (using the *NS3* [7] simulator) and evaluate our proposed algorithm, *CP-DASH*, by comparing it with five other adaptive video streaming techniques [3], [6], [8]–[10]. We demonstrate that our proposed method substantially reduces the switch and stall rates while maintaining a comparable streaming video resolution.

II. Background and Motivation

We begin by presenting a broad discussion of the work on HTTP-based adaptive video streaming. A brief principle of bit-rate adaptation algorithms is given to introduce the contemporary technology and our focused point. Then, we present an overview of frame aggregation and the fundamentals of aggregated control packets.

A. Dynamic Adaptive Video Streaming over HTTP

Dynamic Adaptive Video Streaming over HTTP (DASH) is an adaptive bit-rate streaming technique. DASH is used for streaming media content via an HTTP web server, such as Adobe's HDS and MPEG-DASH. Intrinsically, a video stream is encoded into multiple discrete bit-rate segments or chunks and stored on the server side. According to the perceived network conditions, such as broadcast buffers or throughput, client side attempt to optimize the various indicators that constitute user's QoE, which including video quality, stall rate and stability. The client can switch to a low-quality video version to avoid insufficient traffic during temporary network congestion and switch back to a higher quality after the network conditions improve. For a specific instance, a bit-rate stream is broken into multiple two to ten seconds segments. To provide smoothly video, when bit-rate is decreased to lower resolution at i^{th} seconds, the lower resolution video segment of corresponding time point will be load to buffer. So that, buffer could retention at least one video segment to play continuous video streaming [8]. Finally, a video player fetches each segment independently with smooth video quality while the servers do not have to keep track of any playback session states. Those technology have been widely deployed in commercial systems, including Netflix, Microsoft Smooth Streaming, Apple HTTP Live Streaming (HLS), and HTTP-based Dynamic Adaptive Streaming (DASH) [11].

Algorithms for bit-rate selection in DASH include buffer-based [6], [12] and throughput-based algorithms [10], [13]. Buffer level algorithms indirectly reflect historical throughput and avoids video stalling due to an empty buffer, which leverage the historical average utilization of buffer levels to drive the selection of video bit-rate. In contrast, throughput-based algorithms monitor the incoming bit-rate to select higher-resolution segments. From monitoring throughput directly, the adaptation algorithms could effectively choose the bit-rate levels for future chunks to deliver the highest possible QoE [14]. However, due to most these adaptation algorithms require additional time to calculate the buffer level or use history data to predict throughput, various events can directly decreased user's QoE such as start-up latency, rebufferings or bit-rate changes. Thus, we believe that lightweight monitoring network conditions in time could provide a better quality online video for viewers.

B. Frame Aggregation Overview

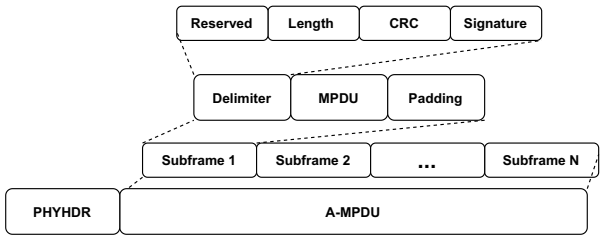

Fig. 1: Examples for frame format of A-MPDU

Frame aggregation was proposed in 802.11e for increasing WiFi throughput by allowing multiple data units with the same destination to be assembled and sent out as one frame. A Block Acknowledgment (Block Ack) is associated with every few aggregated frames which effectively reduces the overhead of the physical header [15]. The aggregated frames can work on two levels that included the MAC service data unit (A-MSDU), and MAC protocol data unit (A-MPDU) aggregation. As a specific example, Fig. 1 illustrates the components of an A-MPDU frame. The A-MSDU scheme can assemble different units (MSDU) into one aggregated frame as sub-frames. These two aggregation mechanisms are operated at different levels, where A-MSDU works on top of MAC layer and A-MPDU works on bottom. The key difference between them is that A-MPDU function could effectively direct the target destination after the MAC header encapsulation process. When available channel width is narrow, more incoming packet units will be aggregated together so that increased each A-MPDU size. In this paper, we focus on the A-MPDU level. Through analyzing the received packet characteristics with the corresponding the send packet, we can get the key characters of the current network conditions, included link congestion.

In this research, we investigate the method of congestion prediction that has the potential approaches for improving user QoE. Meanwhile, we propose our algorithm via congestion prediction in order to provide high-quality and reduce video-stalling. In the next section, we will introduce our proposed method for predicting link congestion and algorithm for selecting video bit-rate.

III. System Design

In this section, we present the Congestion Predication DASH algorithm (CP-DASH) which is buffer-based and estimates link congestion. First, we describe our method for approximating link congestion using frame aggregation. Then,

we present our proposed CP-DASH algorithm that achieves significant improvements in quality by congestion prediction. Finally, a brief description of QoE classes that are used to classify video streaming quality is presented.

A. Link Congestion Prediction

Our method for predicting link congestion is based on observing the A-MPDU Characterization. Fig.2 presents a specific example to illustrate video clients randomly receiving packets from a WiFi link. Then, A-MPDU aggregated same destination packet units and send to the target client buffer. The A_i, B_i, and C_i packets denote the i-th packet to be sent to clients A, B, and C respectively. Due to aggregated frames, all of packets will be sent in order as an A-MPDU, i.e., all of A's packets will be aggregated together and delivered before B and so on. Following different video resolution demands, the size of the chunks are potentially different. Higher resolution chunks require filling more packets as client A in the example. Client B and C represent medium, and lower respectively. When clients are competing for the channel resource, video stalling usually occurs on higher resolutions since the packets cannot be supplied in time. At same time, due to packet latency, more packets are aggregated together in an A-MPDU. Our method is detecting changes in the number of packets unit to predict the current network condition.

The number of packet unit (MPDUs) in each A-MPDU is defined as the **Aggregation Intensity** (AI). Through observing the AI changes, we can potential predict network conditions. The experimental verification is presented at Section IV. Our method for predicting link congestion is based on observing the A-MPDU Characterization. AI_mean is defined as the average value of AI that is extracted from the block acks in a time window ω. We define the aggregation baseline, AI_base, as the condition when no competing traffic is present. The difference between AI_mean and AI_base is defined as the AI change, ΔAI, which can be used to distinguish between the congested and uncongested cases. When the WiFi link is uncongested, most packets can be delivered without delay. However, if the link is in a congested state, then the incoming packets may wait in the queue list. Thus the delay causes more packets to be aggregated and AI will change appropriately. From experimental observations, we find that **AI changes dynamically when the network is congested**. Thus, we assume ΔAI increases over than a certain threshold (Λ) that could infer the current wireless channel is potential congested, described as Equation 1.

$$\Delta AI \begin{cases} >\Lambda & \text{Network Congestion,} \\ <\Lambda & \text{otherwise.} \end{cases} \quad (1)$$

Align-AI Algorithm. We designed an Align-AI Algorithm, presented in Algorithm 1, to find the threshold, Λ. We define N to be the size of the number of bit-rate options, R; Q is the number of the AIs in a time window, ω; and B_i is the buffer occupancy under the current bit-rate, i. Suppose we set the probe link input rate to the first bit-rate R_0 and we assume that there are 10 Block Acks captured in the time window ω. The AIs from these Block Acks are denoted as $(AI_0^0, AI_0^1, AI_0^2, ..., AI_0^9)$, where the subscript denotes the current bit-rate index and the superscript denotes the AI index. The algorithm starts by calculating both the AI_base and the initial AI_mean_0 as the mean of the AIs for the lowest bit-rate, R_0. The algorithm tries to find the threshold Λ by iterating through the bit-rate options. We check if buffer occupancy is safe for next chunk, the bit-rate could be increased until we reach a point where the buffer occupancy is less than the chunk size. This point is where link congestion occurs and we estimate the value of Λ by subtracting AI_base from the corresponding AI_mean.

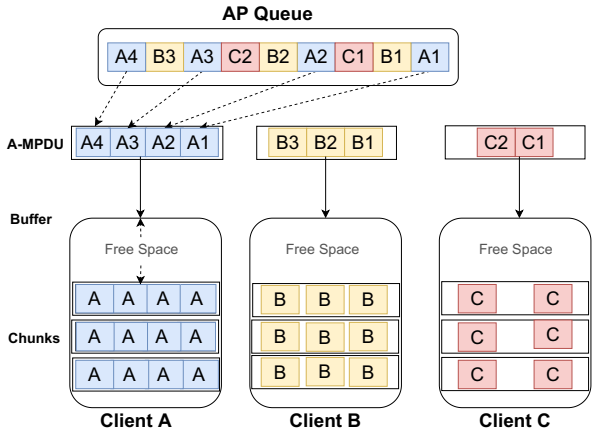

Fig. 2: Examples for scheduling of frame aggregation to client buffer

Algorithm 1 Align-AI Algorithm

Input: R - Bit-rate options (e.g., 2134 Kbps, 2884 Kbps, ...)
N - Size of R
B - Buffer occupancy
ω - Window size
Q - number of AI in ω
Output: Λ - Threshold of AI changed

1: AI_base = AI_mean_0
2: **for** $i = 1$ to N **do**
3: **if** $(B_i - B\_i - 1) < Chunk\_size$ **then**
4: AI_mean_i = $\sum(AI_i^0, AI_i^1, AI_i^2, ..., AI_i^Q)/Q$
5: Λ = AI_mean_i - AI_base
6: **break**
7: **Return** Λ

B. CP: Congestion Prediction DASH Algorithm

Since we convert the problem of estimating congestion into finding the maximum flow without link congestion, we can accurately get the congestion signal via control packet parameter **AI**. Our algorithm leverages congestion prediction and buffer occupancy to provide a stabilized bit-rate selection. The goal of proposed algorithm is to avoid video stalling and dynamic bit-rate switch under the congested WiFi channel. Thus, there are two significant conditions for bit-rate selection:

1) congestion status, observed via AI; 2) buffer occupancy, avoid to run out of buffer.

In our algorithm, the essential dominant parameter is the buffer occupation, which is divided into three zones including *ample*, *normal*, and *danger*. Based on two buffer boundary thresholds; B_{ample} is set as 60% of buffer size; B_{normal} is set as 25% of buffer size; and B_{danger} is set as a single chunk size. Since we do not want to miss any chunk deadline, B_{ample} will keep enough chunks to play videos, B_{normal} provides smoothly bit-rate switch, and B_{danger} avoids video stalling due to run out of the buffer. We defined our Congestion Prediction DASH Algorithm (CP-DASH) shown on the Algorithm 2. For each video chunk, it takes the following metrics as input: cur_rate, the current video bit-rate; Λ, the threshold of AI extracted from Algorithm 1; ΔAI, denotes the values of average AI change during the time window ω; B, the buffer occupancy; R, the set of bit-rate options for all levels of rate R_i(i = 0,1, ..., N). In addition, we need two parameters θ, α to control buffer occupation.

Algorithm 2 CP-DASH Algorithm

Input: cur_rate - Current Video Rate
ΔAI - AI changed (AI_mean - AI_base)
Λ - Threshold of AI changed
B - Buffer occupancy
R - Set of bit-rate options
θ, α - Parameters for controlling buffer size
Output: next_rate, - Next Video Rate

ref = max{ i: $R_i == cur\_rate$ }
if $\Delta AI > \Lambda$ **and** B $> B_{ample}$ **then**
  ref = min{ ref+1, ref_{max}}
else if B $< \theta * B_{ample}$ **and** $ref > ref_{max}/2$ **then**
  ref = min{ ref-1, ref_{min}}
if B $< \alpha * B_{normal}$ **then**
  ref = max{ ref-1, ref_{min}}
  if B $< B_{danger}$ **or** $\Delta AI < \Lambda$
    ref = max{ ref-2, ref_{min}}
else
  ref = max{ ref, cur_rate}
$next\_rate$ = R_{ref}
Return $next\_rate$

As described in Algorithm 2, the reference video rate ref is initialized as index of current video rate. Then, we check the buffer occupancy B and AI change ΔAI. If there is *ample* buffer space and the channel stays in an uncongested level, we increase one level reference video rate. Otherwise, if the buffer level is less than θ percent of B_{ample} and ref is greater than half of rate options, the bit-rate is decreased one level. The reason is because there is the potential to run out of buffer space due to the high bit-rate. The *normal* zone and *danger* zone like Fire-Walls to prevent video stalls and replenish the buffer. At first, if buffer level is less than α percent of B_{normal}, reduce the ref by one level. Once the AI change ΔAI is detected to less than threshold Λ or buffer level is less than B_{danger}, the ref is decreased by two levels. Since the AI change reflected the current channel utilization, if there only are a few channel resources, the bit-rate is requested to rapidly decline. However, to provide smoothing for quality changes in video streaming, each time we only reduce two levels instead of reducing to the minimum. For other cases, to pursue higher quality video, we take a more aggressive method to select the higher rate between ref and cur_rate.

QoE Classes	R_bit	R_switch	R_stall
High	>4Mb/s	<0.2	<0.1
Medium	>2Mb/s	<0.5	<0.4
Low	<2Mb/s	>0.5	>0.4

TABLE I: QoE Classification.

C. QoE Indicators

Previous researches have showed that QoE of video streaming depends on several factors: buffer occupation, bit-rate adoption, download rate, and number of stalling events [8], [9]. In our work, since the key aim of buffer occupation is preventing frequent video stall events caused by running out of buffer space, and over download chunks to wash network bandwidth. The stall rate will be more considered as significant indicator. As well the smoothly of video revealed the switch rate of download bit-rate, which usually has a negative impact on QoE of video streaming. Thus, switch rate is considered as one of QoE indicator in our work. Furthermore, download rate, which is a crucial factor determines the video resolution and quality to provide a better QoE, is also considered as our video indicator. Therefore, we define the QoE classes following three factors (as shown on Eq:2) to indirect the QoE of video: download bit-rate(R_{bit}), switch rate(R_{switch}), and stall rate (R_{stall}).

$$QoE = min\{R_{bit}, R_{switch}, R_{stall}\} \quad (2)$$

To provide high quality experience for video streaming, we believe the lowest plank determines the overall height. The QoE of video is used for classification purposes as the following three classes: “high”, “medium”, and “low”, which are determined by the minimum of these three factors. As shown on Table I, we present the QoE classification with three factors. For bit rate, in order to watch clear and smoothly single video, the clients need a minimum download speed of 2 Megabits per second for 480p resolution videos, where the speed is used to determine the low and median level. Finally, for watching HD video streaming, the clients require at least 4 Mb/s, which is identified as a high QoE level. According to QoE matter analysis [8], if resolution switch rate is over 50% or stall rate over 40%, people will drop or turn off this video. For high quality video, the switch and stall rate will stay lower at least 20% and 10%, respectively.

IV. Experimental Setup and Evaluation

In this section, we will first validate our design of link-congestion prediction. Then, we evaluate the proposed algorithms results with other five state-of-the-art approaches. We focus on evaluate the average bit-rate selection, switch rate and video stalling rate under different network conditions. Since block acks cannot come from free on a smartphone or mobile devices, which requires rooting to be able to “watch” for

block ACKs. Thus, we set up experiments using the *NS3* [7] simulation software, which is based on the platform proposed by Vergados et al. [6].

A. Design Validation

Experimental Verification. According to a broadband report [16], the average internet speed is 35.36 Mbps in the US. Hence, we set up a narrow link channel with 35 Mbps capacity for downloading videos. To easily observe results, we set the bit-rate for all the video clients at 10 Mbps. To investigate how link congestion affects AI, we set up three experiments with 2, 3, and 4 video clients, respectively using TCP. Varying the number of clients helped us observe the different network conditions–uncongested, almost congested, and congested. The traffic flow is generated by sending packets with a fixed rate (5 Mbps) to create an environment with competing clients. To observe video stall under dynamic cross-traffic at WiFi channel, we launched an "On-off" traffic flow that lasting for 10 seconds and off for 10 seconds. To detect the points at which link congestion occurs, we plot AI_mean and stall times for the experiments.

Results Analysis. We can see from Fig. 3 and 6 where the network is uncongested, since the stall rate is less than 5%, and the AI_mean is somewhat stable even with dynamic cross-traffic flow. The reason because there is still ample bandwidth to satisfy all streaming flows. For here, we capture AI_mean with a time window of 20 seconds, which included a complete "On-off" cross-traffic flow roll. When clients increase in Fig. 4, the AI curves fluctuate with values decreasing over certain time periods. Since the cross-traffic competition, client 1 encounters link congestion such that we observe video stalling occurred (between 25 to 125s) for client 1 as shown in Fig. 7. This is because as more cross-traffic comes in, AI will decrease due to increased latency. However, video stall does not occur on clients 2 and 3, because the available resource still can support them with the required input rate. We define this condition as 'almost congested' since the link capacity is exhausted, where the stall rate is less than 40%.

This 'almost congested' condition help us investigate the potential threshold Λ to from the AI change. As shown in Fig. 4. we observed when the video stalling occurs on client 1, its AI_mean is fluctuated. The reason because under a limited channel resource, the dominant link (*i.e.,* the highest link rate) will consume more bandwidth. When the AI_mean for any client reaches a certain deviation from the stable value, it denotes that the link capacity is full. Although at period 20s to 60s, the AI_mean still is consistent since most packets are aggregated at this period. With latency increased at 80s to 140s, the incoming packets are decreased so that AI_mean is decreased in this period. Compared with the AI_mean curves of clients 2 and 3, we assume that if the AI_mean is below than 25 packet units, the congestion has occurred. In this case, the threshold Λ is considered as 25.

To verify this assumption, we conduct experiments with four clients as shown in Fig. 8. Under this condition, all the video clients experience video stall, because the available bandwidth is lower than the required bit-rate for each video client. Furthermore, Fig. 5 shows that the AI_mean for most the video clients is lower than 25. Even AI_mean of client 3 is little higher than 25, the reason it keep a higher AI_mean because most packet are aggregated. The AI_mean for all the clients in the congested case is similar to the AI_mean for client 1 in the 'almost congested' case (as seen in Fig. 4) in that they are all less than 25. These experiments present that we can infer link congestion by observing if ΔAI (i.e. change in AI_mean) goes above a certain threshold. We define this threshold as Λ and describe next how we can determine this threshold for the video clients.

B. Performance Evaluation

Experiment Settings. The goal of the experiment was to simulate a public WiFi area, which investigates multiple clients downloading files from a website and watching the live video streaming at same time. The simulation setup consisted of several nodes including a few server nodes, Web client nodes, video client nodes and one access point node. To simulate background traffic, we required the Web clients, which included both UDP and TCP clients, to request to download or visit the website. For video clients, they try to request download and play a video from the server node using the assigned DASH algorithms. We set up only one AP in the simulator because the cross traffic and interference in the channels have an adverse affect for user QoE. The proposed algorithm aim to maximize the user QoE in this network-crowded area.

In the proposed algorithm, the maximum buffer size was equal to 36 seconds (or equivalently, 18 chunks) and period window size ω for detecting AI was set as 2 sec. The duration of each video chunk was fixed as 2 sec. We provide 8 different video encoding rates from a single video server: 2134, 2884, 3279, 3840, 4220, 4527, 5120, and 5830 Kbps and set constant bit-rate for each chunk. The experiments focus on higher bit-rate because high resolution is a significant component of video QoE. We set α = 0.6 and θ = 0.45 based on empirical observations.

We evaluate the proposed algorithm by comparing its performance with five existing video adaptation algorithms (described below) on the same task sets. We implemented all the algorithms on the *NS3* simulation platform [7].

- **minDash**: In this algorithm, all video segments are assigned the minimum bit-rate.
- **FESTIVE**: This approach uses the harmonic mean of throughput history to guide the bit-rate selection [3]. To avoid bit-rate over-selection, each video segment chooses the highest bit-rate that is less than the average throughput of previous 20 video chunks.
- **F-DASH**: This approach employs fuzzy logic to control the buffering time and resolutions [6]. It distributes the best possible resolution of video segments and delivers undisrupted video playback as a result of buffer underflows at the client.

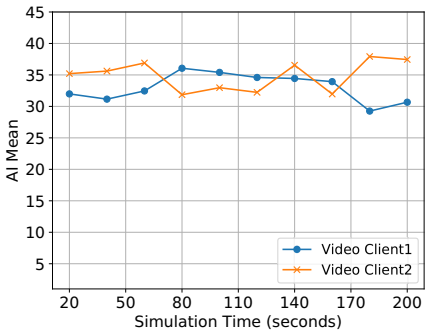

Fig. 3: AI with 2 clients

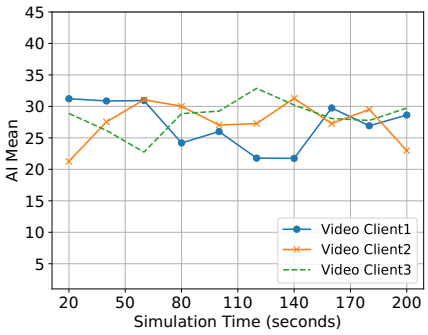

Fig. 4: AI with 3 clients

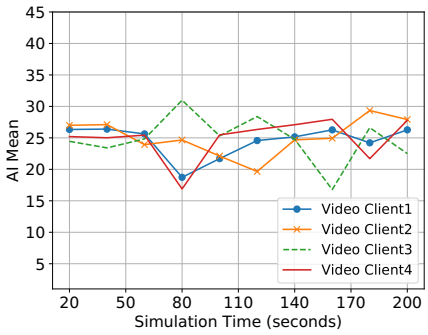

Fig. 5: AI with 4 clients

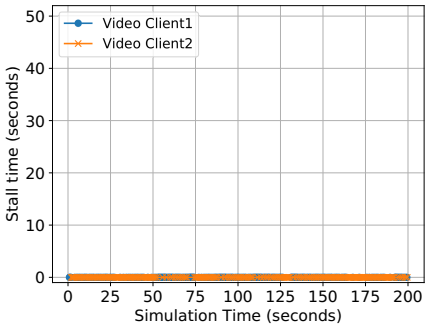

Fig. 6: Video Stall with 2 clients

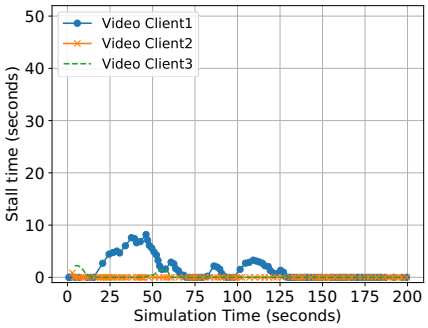

Fig. 7: Video Stall with 3 clients

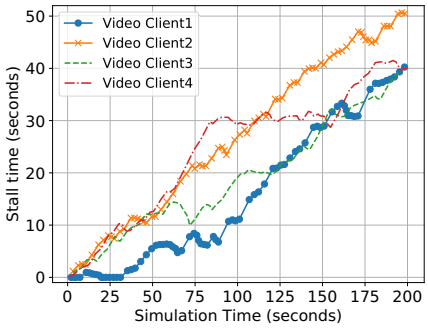

Fig. 8: Video Stall with 4 clients

- **PBA**: This method chooses the highest bit-rate that is less than the predicted bandwidth [9], where it is assumed that available bandwidth is known. To avoid video stalling, the approach sets a buffer-protect mechanism that reduces the bit-rate when buffer storage is less than 30% of the buffer size.
- **TOB2**: This approach considers a boundary for selecting the bit-rate for the next video segment [10]. When the throughput of last incoming segment is greater than a specific percentage (85%) of previous average segment throughput, the bit-rate will be increased. This method includes a buffer-protect mechanism to avoid video stalling.

We evaluate the algorithms under a dynamically cross traffic flows. In this experiment, we start by streaming a 200-second video on a clear WiFi channel whose link capacity is 35 Mbps. For stable competing flows, we simulated Web clients by generating two UDP competing clients and one TCP client in order to simulate different client types. For each web client, we set the flow size at a consistent value of 4 Mbps. To simulate the unstable incoming traffic flows, we generated an "On-off" traffic flow lasting for 10 seconds and then sleeping for 10 seconds recursively. The "On-Off" traffic flow took less than 30% of the link capacity and was selected from a linear function, $flow\_size(\text{“}On-Off\text{”}) = (link\_capacity * 0.3) - (time * 0.1) - \sigma$, where *time* is simulator time and σ denotes a random value between 0 to 2. We evaluated the adaptation algorithms by varying the number of video clients (set at 1, 3, and 5) and each number of clients and adaptation algorithm repeated testing 10 times.

Results Evaluation. We evaluate the experimental performance of the algorithms using three metrics: average video bit-rate, stall rate, and switch rate. *Average video bit-rate* represents the average of the bit-rates selected over the entire video. *Switch rate* measures the smoothness or the frequency at which the bit-rate switches per video segment and *Stall rate* is the percentage of the time when the video stalls.

Figures. 9, 10, and 11 plot the the average video bit-rate, switch rate, and stall rate, respectively, for all the evaluated algorithms and for the different number of competing video clients. In the figures, the histograms denote the means and error bars represent the standard deviations. Ideally, good algorithm performance is determined by *higher* values for average video bit-rate and *lower* values for both switch rate and stall rate.

The *minDash* algorithm denotes the baseline of bit-rate options. In Fig. 9, we can see that our proposed algorithm performs as well as or better than the algorithms with relatively higher average bit-rates for each of the different number of clients. When there is a single video client, there is ample bandwidth available and hence most of the algorithms perform well. Our algorithm has a slow start in each case because it spends time detecting the current network condition via AI. Despite this, our algorithm continues to achieve a relatively higher average bit-rate when the number of clients increases.

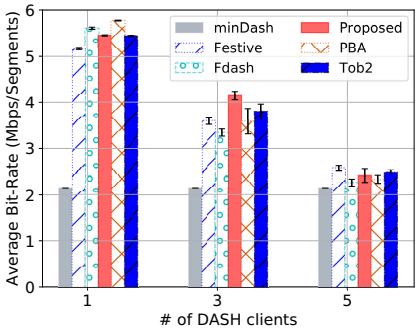

Fig. 9: Average video Bit-rate

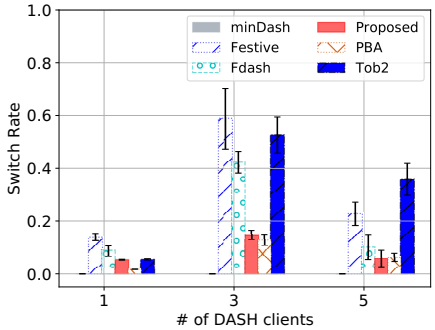

Fig. 10: Switch Rate

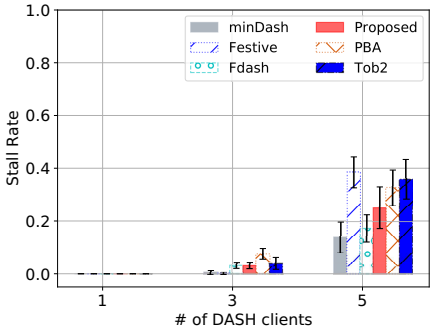

Fig. 11: Stall Rate

The biggest advantage that our algorithm provides is in offering consistently lower values for the switch rate leading to better user QoE, as shown in Fig. 10. When the number of clients is 5, our proposed method's switch rate (0.05) is lower than that of the *Festive* algorithm (0.22) and *Tob2* (0.36) and is comparable to that of *PBA* (0.06). When the number of clients increases, the available bandwidth is much reduced and runs between normal and danger zones leading some of the adaptation algorithms (e.g., *Festive, Fdash, tob2*) to more frequently switch their bit-rates. Since we look at link congestion, which occurs before the bandwidth enters the danger zone, our algorithm switches the bit-rate to a lower value before the bandwidth enters the danger zone and maintains it at this value without requiring frequent switching. The *PBA* algorithm, similarly, leverages available bandwidth information to determine a precise bit-rate without frequent switching. Our algorithm also achieves relatively low values for the stall rate, as shown in Fig. 11. Our algorithm's stall rate (0.25) is better than that of *FESTIVE* (0.38) and *PBA* (0.32) but less than that of *Fdash* (0.17) when there are 5 clients.

Overall, following the QoE classification (Table I), our method provides a good balance of all the three metrics – it achieves relatively higher average bit-rates, low switch rates, as well as low stall rates for the different number of clients, thereby contributing to a good user QoE.

V. Related Work

WiFi Channel Characterization: WiFi Channel Characterization provides valuable information to help resolve various network problems such as link congestion. Passive client monitor methods are cheap and flexible choice for monitoring wireless link performance. [17] leveraged frame aggregation and developed a light-weight method to estimate throughput and airtime of a wireless channel by observing a few control packets. Other prior works[[5]] focus on estimating available bandwidth by relying on physical layer observations. Despite their non-intrusive nature, passive approaches usually suffer from severe loss that lead to inaccurate measurements due to client's limited observation capability

For monitoring current network conditions, Song et al. [18] proposed *AIWC* (Aggregation Intensity based Wifi Characterization), an active measurement method that induces frame aggregation by sending probe trains of different rates and quantifies the response to estimate available bandwidth. While more capable in their observations as compared to passive approaches, such methods also fall short since aggressive WiFi scanning can adversely impact energy and throughput especially in crowded wireless environments [19] .

Live Video Streaming: Many existing adaptation algorithms rely on throughput history to pick a suitable bit-rate [10], [13]. FESTIVE [3], for example, is an adaptation algorithm that uses harmonic mean of the throughput of previous segments and considers the delay between video bit-rate updates to minimize the number of rate switches. Most methods, however, provide inaccurate predictions and incur high cost on the link. In contrast to throughput prediction, estimating the available network bandwidth consumes fewer channel resources and also provides precise observations to adjust encoding rate. Zou *et al.* [9] showed that current streaming adaptation algorithms only achieve 69%-86% of optimal quality. Therefore, there is still more that should be done in terms of improving network performance estimation algorithms that guide rate adaption methods.

VI. Conclusion

In this work, we designed a mechanism to predict congestion leveraging frame aggregation at WiFi network. Based on that, we proposed a video streaming adaptation algorithms (CP-DASH) that focused on reducing video stalling under congested WiFi environment. By exploiting a lightweight passive WiFi channel characterization, we showed our method can effectively predict link congestion on a WiFi channel. For target reducing video stalling rate, we show that during startup, the proposed algorithms promises to deliver a better video quality compared to throughput-based algorithms. Overall, we show that CP-DASH algorithms with different stability functions can trade off stalls and maintain higher bit-rate selection, which much improved average quality. For future work, we continue to address some of the open challenges in bandwidth predictions and other network protocols for video streaming download in real time.

ACKNOWLEDGMENT

We thank all reviewers for their critical comments, and all the participants in the experiments and paper editing. This material is based upon work supported by the National Science Foundation under Grant No.CNS-1718405.

REFERENCES

[1] V. Cisco, "Cisco visual networking index: Forecast and trends, 2017–2022," *White Paper*, vol. 1, 2018.

[2] 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*, vol. 32, no. 4, pp. 719–733, 2014.

[3] 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 (ToN)*, vol. 22, no. 1, pp. 326–340, 2014.

[4] J.-P. Vasseur and S. Dasgupta, "Probing available bandwidth along a network path," Nov. 7 2017, uS Patent 9,813,259.

[5] D. N. da Hora, K. Van Doorselaer, K. Van Oost, R. Teixeira, and C. Diot, "Passive Wi-Fi link capacity estimation on commodity access points," 2016.

[6] D. J. Vergados, A. Michalas, A. Sgora, D. D. Vergados, and P. Chatzimisios, "FDASH: A fuzzy-based MPEG/DASH adaptation algorithm," *IEEE Systems Journal*, vol. 10, no. 2, pp. 859–868, 2015.

[7] NS3, *A discrete-event network simulator for Internet systems*, 2019. [Online]. Available: https://www.nsnam.org/

[8] H. Nam, K.-H. Kim, and H. Schulzrinne, "QoE matters more than QoS: Why people stop watching cat videos," in *IEEE INFOCOM 2016-The 35th Annual IEEE International Conference on Computer Communications*. IEEE, 2016, pp. 1–9.

[9] X. K. Zou, J. Erman, V. Gopalakrishnan, E. Halepovic, R. Jana, X. Jin, J. Rexford, and R. K. Sinha, "Can accurate predictions improve video streaming in cellular networks?" in *Proceedings of the 16th International Workshop on Mobile Computing Systems and Applications*. ACM, 2015, pp. 57–62.

[10] H. Ott, K. Miller, and A. Wolisz, "Simulation Framework for HTTP-Based Adaptive Streaming Applications," in *Proceedings of the Workshop on ns-3*. ACM, 2017, pp. 95–102.

[11] H. Zhu, Y. Cao, W. Wang, B. Liu, and T. Jiang, "QoE-aware resource allocation for adaptive device-to-device video streaming," *IEEE Network*, vol. 29, no. 6, pp. 6–12, 2015.

[12] 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," in *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 4. ACM, 2014, pp. 187–198.

[13] P. Zhao, W. Yu, X. Yang, D. Meng, and L. Wang, "Buffer data-driven adaptation of mobile video streaming over heterogeneous wireless networks," *IEEE Internet of Things Journal*, vol. 5, no. 5, pp. 3430–3441, 2017.

[14] Y. Sun, X. Yin, J. Jiang, V. Sekar, F. Lin, N. Wang, T. Liu, and B. Sinopoli, "CS2P: Improving video bitrate selection and adaptation with data-driven throughput prediction," in *Proceedings of the 2016 ACM SIGCOMM Conference*, 2016, pp. 272–285.

[15] L. Song and A. Striegel, "SEWS: A Channel-Aware Stall-Free WiFi Video Streaming Mechanism," in *Proceedings of the 28th ACM SIGMM Workshop on Network and Operating Systems Support for Digital Audio and Video*. ACM, 2018, pp. 67–72.

[16] B. Team, *US States With the Worst and Best Internet Coverage 2018*, 2018. [Online]. Available: https://broadbandnow.com/report/us-states-internet-coverage-speed-2018/

[17] A. S. L. Song, A. Mohammed, "A Passive Client Side Control Packet-based WiFi Traffic Characterization Mechanism," *IEEE ICC*, 2020.

[18] 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)*. IEEE, 2017, pp. 1–9.

[19] X. Hu, L. Song, D. Van Bruggen, and A. Striegel, "Is there WiFi yet?: How aggressive probe requests deteriorate energy and throughput," in *Proceedings of the 2015 Internet Measurement Conference*. ACM, 2015, pp. 317–323.