Detecting Bottleneck Use of PIE or FQ-CoDel Active Queue Management During DASH-like Content Streaming

Jonathan Kua*, Philip Branch† and Grenville Armitage†‡
*Deakin University, Geelong, Australia
†Swinburne University of Technology, Melbourne, Australia; ‡Netflix Inc
jonathan.kua@deakin.edu.au, pbranch@swin.edu.au, gj.armitage@gmail.com

***Abstract*—Dynamic Adaptive Streaming over HTTP (DASH) is a widely adopted standard for delivering high Quality of Experience (QoE) for consumer video streaming applications. The progressive deployment of Active Queue Management (AQM) schemes – such as PIE and FQ-CoDel – at ISP bottlenecks or home gateways means that consumers' video streams are increasingly impacted by such AQM schemes. However, many existing approaches do not consider adjusting streaming strategies based on the bottleneck queue types. We have previously demonstrated the benefits of AQM schemes for DASH video streams, and proposed adaptive chunklets for an improved streaming performance. In this paper, we demonstrate the problems of queue-agnostic streaming and propose a queue-detection technique during DASH-like streaming. This entirely client-side and application-level technique is capable of detecting likely FIFO, PIE and FQ-CoDel AQM schemes at network bottlenecks.**

***Index Terms*—DASH, TCP, AQM, FIFO, PIE, FQ-CoDel, QoE, bottlenecks, content streaming, chunklets, Coefficient of Variance**

I. Introduction

The past decade has seen a rapid increase in the volume of Internet video traffic. In 2019, video traffic accounted for more than 60% of the total downstream traffic on the Internet[1], with a forecast of it reaching 82% by 2021[2]. The demand for a better Quality of Experience (QoE) has also increased, with users intuitively preferring videos to start quickly, stream at a higher quality with minimal rebuffers/stalls. However, unpredictable network conditions can negatively impact users' QoE. Hence, many streaming companies have adopted Dynamic Adaptive Streaming over HTTP[3] (DASH)-like technologies for content streaming and delivery. DASH aims to provide an uninterrupted, high quality, and the best possible streaming experience under various network conditions. DASH clients use an internal adaptive bitrate (ABR) algorithm to dynamically select an appropriate video encoding rate to match the network capacity in order to provide a smooth streaming experience [1].

In a consumer home environment, DASH-based video streams have to compete with many different application traffic flows for the last-mile ISP bottleneck bandwidth capacity. The combination of using Transmission Control Protocol (TCP) and conventional first-in-first-out (FIFO) buffer management schemes can result in the *bufferbloat* phenomenon [2]. The IETF recently standardised Proportional Integral controller Enhanced (PIE)[4], Controlled Delay (CoDel)[5] and FlowQueue-CoDel (FQ-CoDel)[6] Active Queue Management (AQM) schemes to address the ramifications of bufferbloat.

We have previously studied the benefits of AQM schemes on DASH-like content streaming [3], and proposed 'chunklets' [4] and 'adaptive chunklets' [5] to complement ABR algorithms to improve streaming experience in AQM-enabled environments. These techniques were primarily motivated by the deployment of AQMs at the bottleneck. Our experiments showed that chunklets do not react very well when FIFO buffers are used, which leads to a key research question: How can a DASH streaming client detect the bottleneck use of AQM (in particular, distinguishing between likely FIFO, PIE and FQ-CoDel) so as to enable a more effective streaming strategy?

In this paper, we demonstrate the problems of queue-agnostic streaming and propose an entirely client-side, application-level quantitative measure for detecting likely FIFO, PIE and FQ-CoDel at the bottleneck. Section II reviews background information and related work. Section III describes our evaluation methodology and present experimental results. Section IV concludes and outlines future work.

II. Background and Related Work

In this section, we briefly describe the operations of DASH and chunklets streaming, AQM schemes and related work.

A. DASH and chunklets overview

DASH video content is pre-encoded into multiple versions at different discrete encoding bitrates, or Representation Rates (RR). Each encoded video is then segmented into small multi-second video segments/chunks. All video chunks encoded at different bitrates are aligned in the video timeline, so that the streaming client can seamlessly switch between video bitrates at the chunk boundary if necessary. The video server is a standard HTTP web server that provides a corresponding manifest file which specifies the information of the available

[1]Sandvine Global Internet Phenomena Report 2019
[2]Cisco Visual Networking Index: Forecast and Trends, 2017–2022
[3]https://www.iso.org/standard/57623.html
[4]https://tools.ietf.org/html/rfc8033
[5]https://tools.ietf.org/html/rfc8289
[6]https://tools.ietf.org/html/rfc8290

video content. DASH streaming clients use an internal adaptive bitrate (ABR) algorithm to adapt to fluctuating network conditions. ABR algorithms use various feedback signals observed for each chunk to determine the RR of the next chunk to be retrieved, *e.g.* recently achieved per-chunk throughput *a.k.a* Achieved Rates (AR) and/or playout buffer occupancy [1].

In [4], [5] we proposed 'chunklets' and 'adaptive chunklets' to improve streaming QoE in AQM-enabled environments. Chunklets effectively provide a "larger perceived bandwidth pipe" which influences the client's ABR algorithm to select higher RRs – the "Achieved Rate Multiplication Effect". Adaptive chunklets is a novel enhancement that ensures chunklet-enabled clients are (relatively) fair to other flows sharing the same bottleneck while maintaining high QoE.

B. PIE, CoDel and FQ-CoDel AQM overview

AQM schemes are deployed to reduce queuing delays at network bottlenecks. PIE infers queuing delays from instantaneous queue occupancy and queue egress rate. PIE allows packets arriving within the first 150ms of an empty queue to pass through. Subsequent packets are randomly dropped based on a calculated probability. This probability is adjusted periodically (30ms by default) based on the difference between the current estimated queuing delay and the pre-configured target delay of 15ms. If the drop probability is less than 10%, packets from flows that enabled Explicit Congestion Notification (ECN) will be marked instead of being dropped.

CoDel detects standing queues by measuring the local minimum queuing delay experienced by packets within a specified interval (initial value of 100ms). It calculates the queuing delay by time-stamping packets upon their arrival and departure. If the minimum queuing delay is less than the target delay of 5ms, packets are neither dropped nor ECN-marked. If it exceeds the target delay for at least one interval, CoDel starts dropping packets and sets the next drop time. This drop time is set based on a control law. When the queuing delay is below the target delay again, CoDel stops dropping packets.

FQ-CoDel shares capacity evenly by first classifying incoming traffic flows into one of 1024 sub-queues by hashing the flow's five-tuple. Each sub-queue is then separately managed by the CoDel algorithm and served by a Deficit Round Robin (DRR) scheduler. During each iteration, the scheduler will dequeue up to a 'quantum' of bytes (1500 bytes by default), indirectly allowing FQ-CoDel to prioritise queues with packets from new traffic flows or from 'sparse' traffic flows.

C. Related work

Many studies have shown the benefits of AQMs on various application traffic flows [3], [6]. Knowing whether an AQM is present at the bottleneck can help improve network designs and transport protocol behaviours. There is currently limited literature on AQM detection, especially on detecting more recent schemes such as PIE and FQ-CoDel. In [7] the authors developed an active measurement tool to detect PIE, CoDel and Adaptive RED AQMs deployed along the network path by tracking queuing delays trends and packet losses. Low-priority and delay-sensitive transport algorithms typically do not perform well in the presence of AQMs. Alternative Backoff (ABE)[7] is a sender-side TCP modification that allows TCP to back off less aggressively when detecting the presence of AQMs (using ECN marking as indicator). Algorithms such as TCP Prague [8] also implemented mechanisms to react differently upon detecting the use of AQMs at the bottleneck.

In this paper, we present an entirely client-side, application-level method that does not require any operating system modifications to the sender, receiver, or in-network devices.

III. Evaluation Methodology and Results

In this section, we describe our evaluation methodology and present our experimental results.

A. Experimental testbed setup and performance metrics

We use the same experimental testbed setup and performance metrics as described in [5]. In this work, we use *intra-chunk* (chunklets) and *inter-chunk* (chunk) measurements of AR variations to detect bottleneck AQM types. For intra-chunk measurements, we instrumented a tool to measure the AR experienced by each chunklet within a chunklet cluster. We then use the *Coefficient of Variation* (CV, *a.k.a.* relative standard deviation) to measure the variability of per-chunklet AR values within a cluster across the whole streaming session. For inter-chunk measurements, we also use CV to measure the variability as experienced by each chunk across the whole streaming duration. CV typically measures the dispersion of a probability or frequency distribution, and is commonly defined as the ratio of standard deviation (σ) to the mean (μ): $CV = \frac{\sigma}{\mu}$

B. Experimental results and analysis

1) Summary of results: We reproduced the experimental results published in [5]. These experiments involved a chunklet-enabled DASH client competing with $M = \{2, 4, 6, 8, 10\}$ elastic TCP flows using $N = \{1, ..., 10\}$ chunklets across a shared 12/1Mbps bottleneck managed by {FIFO, PIE, FQ-CoDel} schemes. Due to space constraints, we refer the reader to [5] for a comprehensive set of results and analysis.

2) Potential implications of queue-agnostic approaches: A large number of chunklets across FIFO leads to high instability index (a higher frequency and larger magnitude of RR switches) which can be detrimental to users' QoE. Figures 3, 4, 5 present illustrative results from a subset of experiments when N={4, 10} chunklets compete with M=8 elastic TCP flows across FIFO, PIE, and FQ-CoDel bottlenecks.

Figure 3 represents illustration of a typical problem. As N increases, chunklets flows are increasingly correlated, essentially self-competing in addition to competing with other elastic flows. This can cause highly-fluctuating ARs which in turn results in the selection of varying RRs, leading a high instability index. Users might favour N=4 over N=10 chunklets where RRs are more stable throughout the streaming session.

[7]https://tools.ietf.org/html/rfc8511

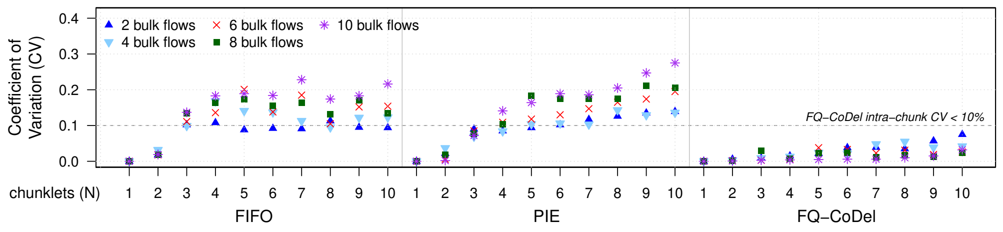


Fig. 1. Intra-chunk CV (median) for $N = \{1, ..., 10\}$ chunklets and $M = \{2, 4, 6, 8, 10\}$ competing elastic flows over FIFO, PIE and FQ-CoDel.

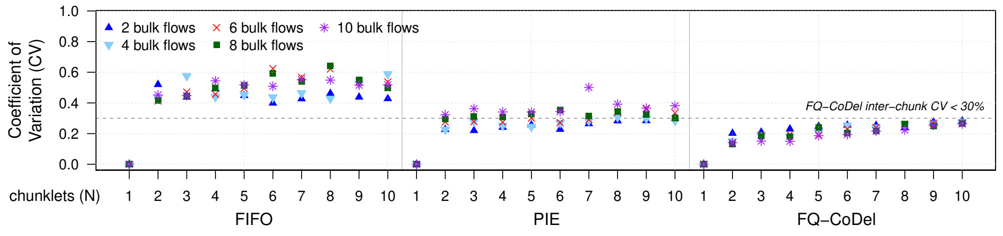
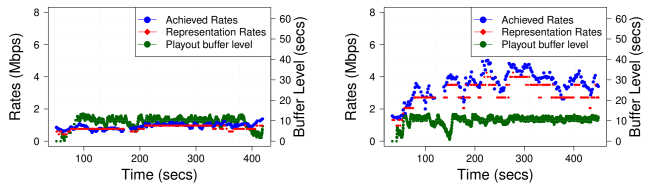
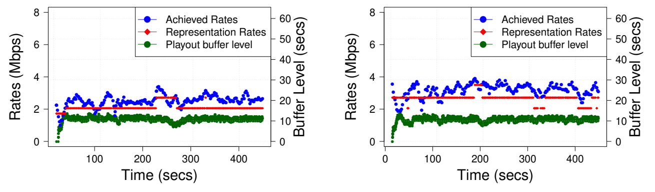


Fig. 2. Inter-chunk CV values of AR for $N = \{1, ..., 10\}$ chunklets and $M = \{2, 4, 6, 8, 10\}$ competing elastic flows over FIFO, PIE and FQ-CoDel.



(a) N=4 (b) N=10

Fig. 3. AR, RR, playout buffer vs time, M=8, FIFO



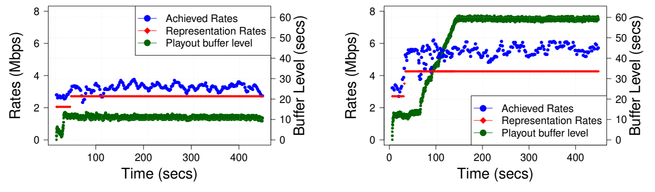
(a) N=4 (b) N=10

Fig. 4. AR, RR, playout buffer vs time, M=8, PIE

AQMs alleviate the problem of flow correlation by actively inducing packet losses well before the queue is full. Figures 4 and 5 demonstrate how both PIE and FQ-CoDel provide a much better experience than FIFO, with FQ-CoDel being more superior. In this instance, users would intuitively prefer N=10 to N=4, where higher RRs are streamed throughout the session and with minimal RR switches, *i.e.* lower instability index.

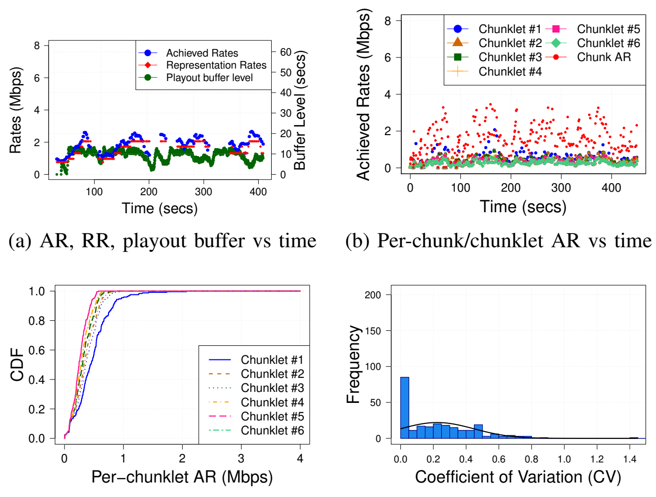
3) Detecting PIE and FQ-CoDel using intra and inter-chunk measurements : Accounting for the potential implications of queue-agnostic streaming, we implemented a queue detection technique entirely within the client at the application level. We explore the use of per-chunklet (intra-chunk) and per-chunk (inter-chunk) AR values measured by the client to determine the bottleneck queue type. We use CV (as described in Section III-A) as a metric to evaluate the correlation properties and distributions of intra and inter-chunk AR values. Section III-B4 first presents a subset of the experimental results for illustrative purposes. In these experiments, N=6 chunklets compete with $M = 10$ elastic TCP flows. Section III-B5 then presents the aggregated results across all experiments.

4) Time series representation : Each of the following figures (Figures 6, 7 and 8) comprises the following subfigures: (a) Per-chunk AR values, RR and playout buffer level vs time; (b) Per-chunklet AR values vs time overlaid with per-chunk AR values; (c) Cumulative Distribution Function (CDF) representing the distribution of per-chunklet AR values; and (d) Histogram representing the intra-chunk CV distribution for all chunklet clusters throughout a streaming session.



(a) N=4 (b) N=10

Fig. 5. AR, RR, playout buffer vs time, M=8, FQ-CoDel

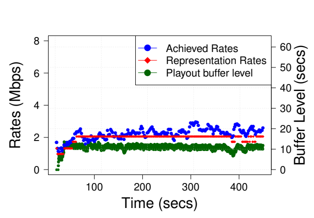


(a) AR, RR, playout buffer vs time (b) Per-chunk/chunklet AR vs time

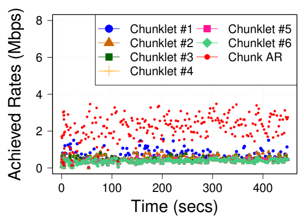
(c) Per-chunklet AR CDF (d) CV distribution

Fig. 6. N=6 chunklets competing with M=10 flows across FIFO

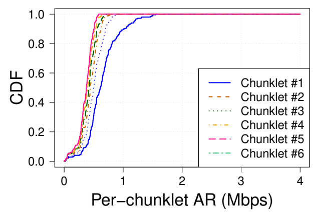
Figures 6 and 7 show that FIFO and PIE manifested similar behaviours in terms of per-chunklet AR values and CV distribution (with PIE consistently increasing as N increases, *cf.* Figure 1). The distinctive properties of FQ-based AQMs (in this case, FQ-CoDel) are clearly reflected in the intra-chunk CV measurements, as shown in Figure 8. Figure 8d shows a CV distribution plot that is highly right-skewed, showing a very tight intra-chunk AR distributions – a strong indicator of the presence of FQ-based AQM along the network path.



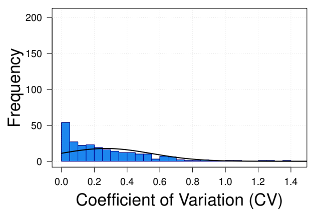
(a) AR, RR, playout buffer vs time



(b) Per-chunk/chunklet AR vs time

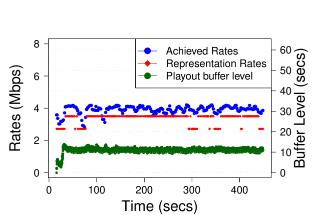


(c) Per-chunklet AR CDF

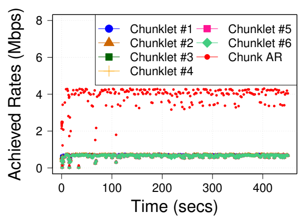


(d) CV distribution

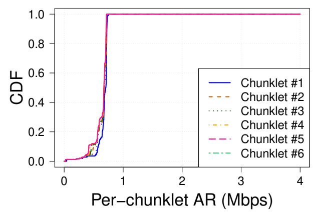
Fig. 7. N=6 chunklets competing with M=10 flows across PIE



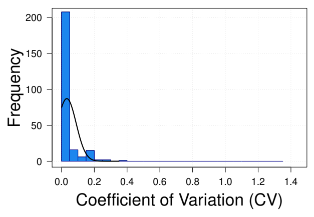
(a) AR, RR, playout buffer vs time



(b) Per-chunk/chunklet AR vs time



(c) Per-chunklet AR CDF



(d) CV distribution

Fig. 8. N=6 chunklets competing with M=10 flows across FQ-CoDel

Therefore, we can conclude that the tighter or closer the distribution (tending towards zero, histogram plot skewing to the right), the more FlowQueue-like the bottleneck; the larger the spread, the more single queue-like the bottleneck.

5) Aggregate CV distributions : We have seen how CV can provide a normalised and quantifiable way of characterising queue type. Figures 1 and 2 show the intra and inter-chunk CV values for all experiments. In Figure 1, the median intra-chunk CV values for all queuing schemes increase as N and M increase. FIFO behaves as expected with noisy, least consistent variations as both N and M increases. PIE shows relatively high CV values but follows a consistent pattern (potentially differentiates PIE from FIFO). FQ-CoDel demonstrates the best performance by consistently achieving median intra-chunk CV $<$ 10% even when N and M values are high.

Inter-chunk CV measurements provide a clearer dimension for detecting bottleneck queue type. Figure 2 shows FQ-CoDel consistently maintains a CV value $<$ 30% throughout the streaming session (due to FQ enforcing even-capacity sharing), PIE keeps the variation between 20-30% (due to active probabilistic packet drops resulting in long-term relatively even capacity sharing), whereas FIFO's variation is highly inconsistent as expected. Our analysis shows that FQ-CoDel has the least intra-chunk variations with $\mu <$ 10% and inter-chunk variations as indicated by $\sim$ 20% CV in all scenarios.

IV. Conclusions and Future Work

In this paper, we experimentally demonstrated the potential problems of queue-agnostic streaming and proposed an entirely client-side, application-level quantitative measure for detecting bottleneck PIE and FQ-CoDel AQM schemes. We show that by using both intra-chunk and inter-chunk Coefficient of Variation measurements, we can detect queuing algorithms that are deployed at the bottleneck – distinguishing AQM schemes from conventional FIFO, and accurately identify if a bottleneck deploys FlowQueue-like AQM algorithms.

Future work includes integrating queue-detection signals into adaptive chunklets and ABR algorithms, characterising streaming behaviours over other emerging AQM schemes and performing experiments in-the-wild (over the public Internet). Other interesting avenues include evaluating the performance of video clients communicating across network paths involving moving bottlenecks, and applying machine learning-based methods (such as feature engineering approaches) for classifying/detecting AQM schemes deployed at network bottlenecks.

References

[1] J. Kua, G. Armitage, and P. Branch, "A Survey of Rate Adaptation Techniques for Dynamic Adaptive Streaming over HTTP," *IEEE Communications Surveys & Tutorials*, vol. 19, no. 3, pp. 1842–1866, 2017.

[2] J. Gettys and K. Nichols, "Bufferbloat: Dark Buffers in the Internet," *Queue*, vol. 9, no. 11, pp. 40:40–40:54, Nov. 2011.

[3] J. Kua, G. Armitage, and P. Branch, "The Impact of Active Queue Management on DASH-based Content Delivery," in *2016 IEEE 41st Conference on Local Computer Networks*, Nov 2016, pp. 121–128.

[4] J. Kua and G. Armitage, "Optimising DASH over AQM-enabled Gateways Using Intra-Chunk Parallel Retrieval (Chunklets)," in *2017 26th International Conference on Computer Communication and Networks (ICCCN)*, 2017, pp. 1–9.

[5] J. Kua, G. Armitage, P. Branch, and J. But, "Adaptive Chunklets and AQM for Higher-Performance Content Streaming," *ACM Trans. Multimedia Comput. Commun. Appl.*, vol. 15, no. 4, Dec. 2019.

[6] J. Kua, S. H. Nguyen, G. Armitage, and P. Branch, "Using Active Queue Management to Assist IoT Application Flows in Home Broadband Networks," *IEEE Internet of Things Journal*, vol. 4, no. 5, pp. 1399–1407, Oct 2017.

[7] M. Kargar Bideh, A. Petlund, C. Griwodz, I. Ahmed, R. behjati, A. Brunstrom, and S. Alfredsson, "TADA: An Active Measurement Tool for Automatic Detection of AQM," in *Proceedings of the 9th EAI International Conference on Performance Evaluation Methodologies and Tools*, Brussels, BEL, 2016, p. 55–60.

[8] B. Briscoe and A. S. Ahmed, "TCP Prague Fall-back on Detection of a Classic ECN AQM," Technical Report TR-BB-2019-002, Apr. 2020.