Does TCP's New Congestion Window Validation Improve HTTP Adaptive Streaming Performance?

Anonymous Author(s)

ABSTRACT

HTTP adaptive streaming video flows exhibit on-off behaviour, with frequent idle periods, which can interact poorly with TCP's congestion control algorithms. New congestion window validation (New CWV) modifies TCP to allow senders to restart more quickly after certain idle periods. While previous work has shown that New CWV can improve *transport* performance for streaming video, it remains to demonstrate that this translates to improved *application* level performance, in terms of playback stability. In this paper, we show that enabling New CWV can reduce video re-buffering events by up to 4%, and limit representation switches by 12%, without any changes to existing rate adaptation algorithms.

ACM Reference Format:
Anonymous Author(s). 2022. Does TCP's New Congestion Window Validation Improve HTTP Adaptive Streaming Performance?. In *Proceedings of NOSSDAV '22.* ACM, New York, NY, USA, 7 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Video streaming over HTTP is commonplace, and comprises the majority of Internet traffic [30]. Performance of HTTP adaptive streaming is generally good, and gives a high-quality user experience.

There are, however, some scenarios where HTTP adaptive streaming performs poorly [19, 33]. In particular, the interaction between the on-off traffic patterns generated by chunked streaming applications and TCP congestion control algorithms can reduce the performance of throughput-based video rate adaptation schemes [6, 34]. In some cases, this is due to TCP's congestion window validation (CWV) [28] algorithm, which, while preventing TCP clients from sending using stale knowledge of the network, has been shown to negatively impact the throughput of rate-limited applications [25], including HTTP adaptive streaming. New congestion window validation (New CWV) [13] has been proposed to address this. Prior work [25] has demonstrated that New CWV has the desired *transport* layer impact, but it remains to show that this translates to improved quality of experience (QoE) performance at the *application* layer. This is not guaranteed, given the complexity that exists at both layers, and that results from their interaction. For example, large discrepancies between the video's bandwidth requirements and the available link capacity, or the requirement for stable, long-lived connections in modern streaming video players (e.g., dash.js [11]), can influence rate adaptation [32].

In this paper, we investigate whether enabling New CWV improves video playback stability, and more generally, improves video QoE. To test our hypothesis, we compare two video streams using TCP New Reno, one with CWV and with New CWV. We collect standard video performance metrics, including bit-rate oscillation, and stall time, to measure stability and QoE. Further, to quantify the impact of New CWV with respect to the inferred network state at the client, we also record the immediate and smoothened client's current link capacity estimations for each delivered video chunk.

In particular, we make the following contributions: (i) an implementation of New CWV for Linux (kernel version 5.4) [1]; (ii) a testbed setup for evaluating New CWV's application layer impact; and (iii) results that demonstrate that New CWV improves video stability, with a 12% reduction in bit rate switches, and a 4% reduction in rebuffering time.

To the best of our knowledge, this is the first paper that studies New CWV's application layer impact. Nazir et al. [25] demonstrated New CWV's effect on the transport layer: we validate their results in §3.2. There has been a large amount of work that has proposed new application layer rate adaptation algorithms [15, 24, 37]. In contrast, we only change the transport algorithm and leave the application as is, studying the transport's impact on the application. Improving performance via transport layer modifications could allow for simpler rate adaptation algorithms at the application layer.

We structure the remainder of this paper as follows. In Section 2, we introduce TCP congestion window validation, including its limitations with respect to HTTP adaptive video flows, before describing New CWV. Section 3 describes our

NOSSDAV '22, June 17, 2022, Athlone, Ireland
© 2022 Association for Computing Machinery.
ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

[1]The code used in this paper will be made available with the camera-ready version.

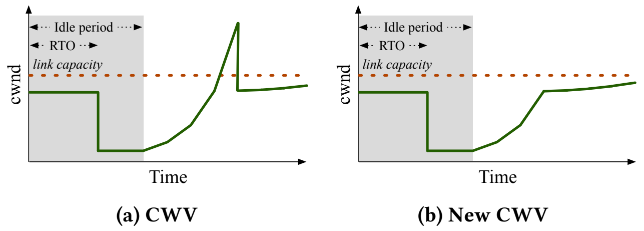


Figure 1: Illustration of *cwnd* growth following an idle period

experimental setup, and the transport and application layer impact of enabling New CWV. Section 4 describes related work, and Section 5 concludes.

2 CONGESTION WINDOW VALIDATION

In HTTP adaptive streaming, a server provides pre-encoded video chunks in different representations, each encoded at multiple bit rates, while the client, using a rate adaptation algorithm, determines the best representation to request at any given time. The goal of the client is to maximise QoE within the network's capacity. This can be a challenge since different, often contradictory, QoE heuristics need be considered simultaneously [31].

Throughput-based rate adaptation algorithms for HTTP adaptive streaming use an estimate of the current network conditions to determine the representation that should be requested. These algorithms require a stable and accurate throughput estimate in order to perform well. However, the interaction between the on-off traffic pattern of streaming video and TCP's congestion control algorithm can lead to significant fluctuations in throughput, impacting the performance of throughput-based rate adaptation algorithms.

In particular, during the idle periods in video transmission between chunks, the TCP congestion controller's knowledge of the network capacity becomes stale. To avoid sending with a possibly unrepresentative congestion window, the congestion window validation [28] (CWV) algorithm resets the TCP congestion window (*cwnd*) to its initial value and forces the connection to re-enter slow-start after an idle period. Figure 1a illustrates the behaviour of CWV following an idle period. CWV has become standard practice [8], and is enabled by default in the latest stable Linux kernel (5.4). However, re-entering slow-start like this, results in packet loss once the CWND grows beyond the link capacity (Figure 2a). This was found to not interact well with HTTP adaptive streaming and other application-limited transmissions [12].

To address this, new congestion window validation [13] has been proposed. One of the modifications in New CWV is that rather than relying on slow-start until packet loss to

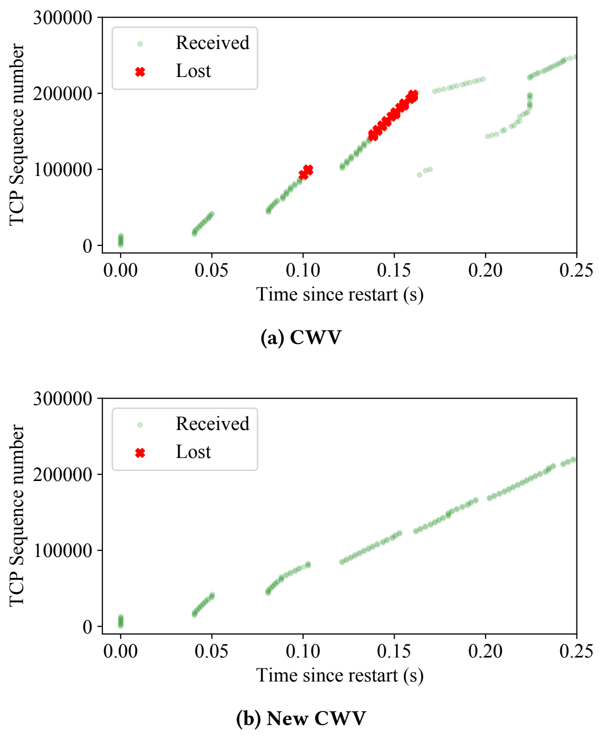


Figure 2: Resumption after an idle period

re-discover an appropriate *cwnd* value after an idle period, New CWV preserves the *cwnd* before the idle period as its slow-start threshold (*ssthresh*), i.e., it uses that value to later exit the slow-start phase. Figure 1b shows the growth of *cwnd* following an idle period under New CWV.

To evaluate the transport performance of New CWV, we have implemented the algorithm within the Linux kernel. We used [3] as a base, which provides a Kernel 3.18 implementation. Our implementation altered two files adding 143 and removing 49 lines of code. We use this implementation to better illustrated the impact of New CWV on flows restarting after an idle period, as shown in Figure 2. As shown, the connection using New CWV uses the previously set *ssthresh* value and leaves slow-start early. This results in New CWV connections not experiencing any packet loss (Figure 2b), after reaching their set *ssthresh* value in the third flight of packets after restarting. In contrast, if the same connection used CWV, the senders would not have preserved the *ssthresh* value that way and would rely on loss to exit slow-start, as seen at the end of the third and fourth flights of packets in Figure 2a. In the presented case, CWV enters congestion avoidance around 160 milliseconds after the transmission

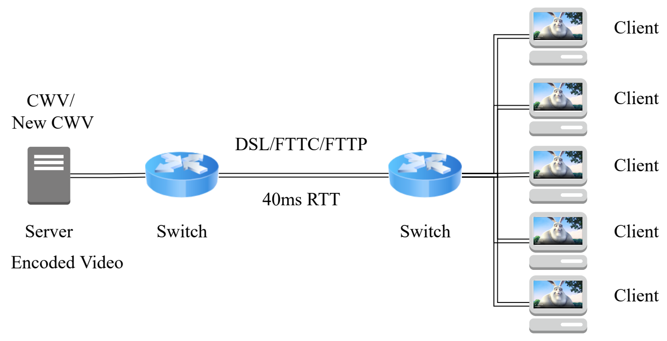


Figure 3: Experimental Setup

restart (Figure 2a), while New CWV is able to enter congestion avoidance around the 110th millisecond mark (Figure 2b).

Overall, New CWV results in fewer lost packets, and returns to its previous sending rate without overshoot after loss, giving more predictable transmission.

New CWV has previously been shown to improve the *transport* layer performance of rate-limited applications when compared with CWV [25], and our implementation and the results we have presented here validate that. It remains to show how this translates into *application* layer performance, particularly since this is not guaranteed [33]. In Section 3, we first validate the results of Nazir et al. [25], before testing our hypothesis that New CWV will enable applications to obtain more consistent throughput estimates, and, in turn, improve the stability of throughput-based rate adaptation algorithms.

3 EVALUATING NEW CWV FOR VIDEO

We first describe our experimental setup (§3.1), which we use to investigate the transport layer impact, verify whether New CWV connections obtain more consistent throughput estimates (§3.2), and later to also investigate the application impact, and more specifically, the difference on video QoE that New CWV connections observe (§3.3). Finally, we summarise our findings (§3.4).

3.1 Experimental Setup

Our evaluation testbed consists of a network emulated in Mininet, running on Ubuntu 20.04, as shown in Figure 3. Both the server and its clients use TCP New Reno, widely used for video delivery [23]. In addition, both are running a modified Linux Kernel (5.4.0). The modifications include a version of New CWV ported to that kernel, alongside RFC 3339 [26] compliant timestamps, to enable better event tracking with higher timing precision. `tcpdump` is used to allow network activity to be reconstructed, which we use to study the packet loss.

The server uses `nginx` (version 1.18) with HTTP/2 delivery enabled. The server provides three representations of Big Buck Bunny [1], encoded at 480p (requiring bandwidth of 0.44Mbps), 720p (2.64Mbps), and 1080p (4.82Mbps). Each representation is provided in chunks that are 3 seconds in duration.

Each client uses Firefox (version 91) with the dash.js (version 4.0.0) player. While the current state-of-the-art rate adaptation algorithm is DYNAMIC [32], we opt to use the throughput algorithm for our experiments. DYNAMIC combines the throughput algorithm with an enhanced version of the BOLA algorithm [33]. By using the throughput algorithm, our evaluations focus on the impact that the transport layer has on throughput estimation at the application layer. Our findings are applicable to the DYNAMIC algorithm, especially in scenarios where the throughput component of the algorithm is used (e.g., in low-latency, live streaming applications).

The network is configured with a bottleneck RTT of 40ms, a reasonable value to emulate connections within a country or region. The routers' queues are sized to the bandwidth delay product. Three different bandwidth profiles are evaluated, representing DSLv2 (10Mbps), FTTC (50Mbps), and FTTP (145Mbps) links; e.g., as are typical in the UK [5]. Below we show the results for the DSLv2 and FTTC links. However, as higher resolution video and other network-heavy operations such as virtual reality environments become more available we expect the issues observed could translate to the links with higher capacity (i.e. FTTC and FTTP).

To evaluate the impact of congestion and competing flows, each simulation was run with multiple clients (1, 2, 3, and 5 clients) simultaneously requesting video. Finally, to reduce noise, we ran each combination of CWV or New CWV, number of clients, and link type, 10 times before reporting the average results. The results presented includes data accumulated from 240 simulations (*2 algorithms* $\times$ *3 link types* $\times$ *4 client variations* $\times$ *10 repetitions*).

During each run we collect the client's bandwidth estimations. Additionally, to evaluate the video QoE impact we collect information to report the rebuffer ratio and the bitrate switch frequency distribution.

3.2 Impact on Transport Performance

New CWV alters TCP's *cwnd* sizing behaviour, allowing it to recover more quickly after an idle period in an active TCP connection. As shown in Figures 1b and 2b, New CWV avoids the packet loss associated with CWV, and we therefore expect clients to report more stable available link bandwidth estimates.

To evaluate this hypothesis, we collect the client's "instantaneous" and "smoothed" bandwidth estimates. The "instantaneous" estimate is obtained by dividing the size of the chunk, in bytes, by the time taken to download it. The "smoothed" estimate takes the "instantaneous" estimate, but combines it with other factors, including historical measurement data, and "safety" or dampening factors. In short, the former is the throughput measurement as seen by the endpoint, while the latter is the input value to the client's rate adaptation algorithm.

Figure 4 shows the instantaneous and smoothed available throughput as calculated by the clients. In particular, all FTTC scenarios (Figure 4b), New CWV has a steeper gradient. This indicates that the throughput estimates are more consistent, falling within a tighter range of values. Since clients are able to leave slow-start earlier, *cwnd* oscillates less when compared to clients using CWV. In addition to being more consistent, clients with New CWV enabled reported estimates that were lower overall. This can be explained by the behaviour of CWV illustrated in Figure 1a: CWV will always reach the maximum link capacity because of its longer slow-start phase. These findings confirm the results reported by Nazir et al. [25], and support our initial hypothesis that the streaming clients measuring throughput will be able to obtain estimates that are more stable.

We illustrate the impact of New CWV on packet loss rates in Figure 5. New CWV consistently achieves lower loss rates when compared to CWV, with New CWV connections having packet loss rates that are up to half that of CWV connections. As explained in Section 2, New CWV exits slow-start earlier, does not overshoot its window, and therefore is able to avoid the loss seen near the end of slow-start that CWV experiences (Figure 2). Nazir et al. [25] observed similar loss values both when New CWV is enabled and when it is not; we believe this due to the much larger RTT value they used.

3.3 Impact on Video QoE

To evaluate whether the improved transport layer performance of New CWV translates into improved QoE at the application layer, we report results for the rebuffer ratio and the bitrate switch frequency. We report results for the DSL evaluations, as this link type, in combination with the video encodings used, best highlights the scenario in which New CWV is most beneficial. With FTTP the combination of clients and video encodings enabled all clients to stream at the highest quality without hitting the link limits. This was also the case for all FTTC simulations, with the exception of the 5 client one. In that scenario, we observed a similar pattern as that shown for DSL. We believe that the problem we describe here is applicable to faster links as network demands increase, with higher resolutions being widely used or as other network-heavy applications start competing for the links' resources (e.g., virtual and augmented reality).

Figure 6 shows the observed bitrate switch distribution. The proportion of the requested chunks (*y axis*) is shown using a logarithmic scale. The representation change (*x axis*) is the magnitude of the chunk-by-chunk bit rate switches. For example, if a chunk is requested at the same bit rate as the preceding chunk, the difference is zero. If the chunk is of the next higher encoding, compared to its predecessor, the difference is 1, and if it is of the next lower encoding the difference is -1, and so on. We see that for the case with 5 clients, connections without New CWV experience non-zero quality switches for 17.4% of the video duration, compared to 5.7% for connections with New CWV. For a video consisting of just over 200 chunks, an 11.7 percentage point difference means that, on average, New CWV connections see 24 fewer switches over the course of the video playback.

Figure 7 shows the percentage that the of the whole video playback that the connection spent in rebuffering state, where the video player stalled, and playback did not progress. Again, looking at the 5 client case, we see that New CWV experiences less rebuffering overall. The mean rebuffering values for the 5 client case are 5% and 1.5% for CWV and New CWV respectively. For a 10 minute video, this 3.5 percentage point difference accounts for over 21 seconds of rebuffering time.

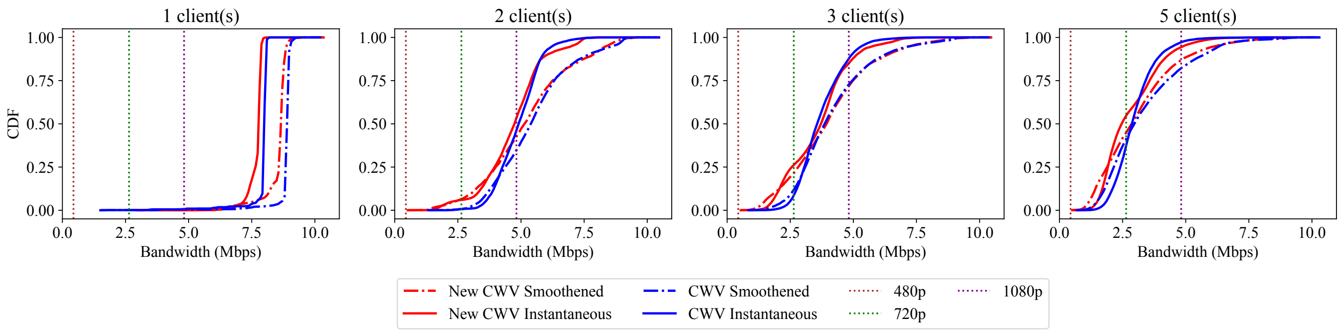
We conclude that New CWV achieves higher video stability when multiple clients are competing on a constrained link, with improved encoding stability and decreased rebuffering time. We have observed this behaviour mainly on the simulated DSL link, since our highest video encoding is just under 5Mbps, however, we note that in practice higher encodings are also used [4].

3.4 Summary

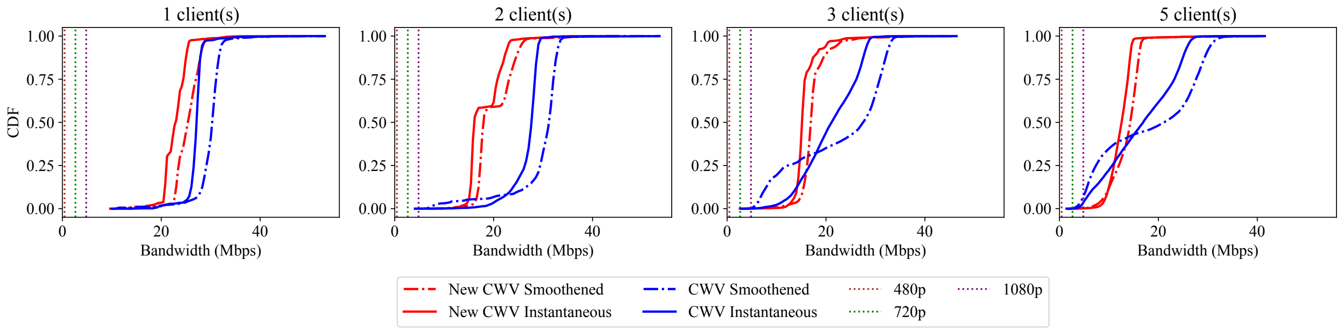
We have validated the throughput results observed by Nazir et al. [25] (Figure 4), but also observed higher packet loss where New CWV is not enabled (Figure 5). Furthermore, our results demonstrate that New CWV's more consistent bandwidth measurements (Figure 4) translate to fewer representation switches (Figure 6). We also conclude that the more consistent measurements better match the available network conditions, allowing clients using New CWV to better adapt to the constraints of the bottleneck link, and thus to request chunks that can be delivered on time without need of rebuffering (Figure 7).

4 RELATED WORK

New CWV enhances the CWV algorithm [28]. Both CWV and New CWV attempt to solve the issue of connection resumption after an idle period in which TCP's view of the network has become stale. While CWV addresses the issue



(a) DSL



(b) FTTC

Figure 4: dash.js client throughput measurements

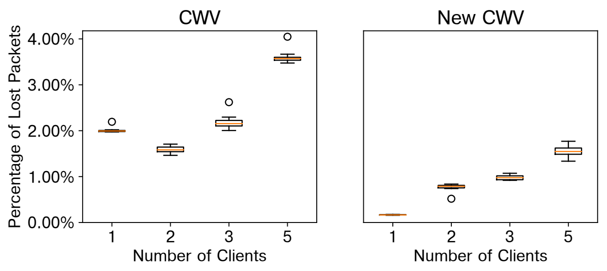


Figure 5: Lost Packets DSL

for bulk, network-limited applications, New CWV improves the algorithm for rate-limited applications. As a result of their changes, both algorithms alter TCP's send rate. Altering the send-rate of TCP should be taken with caution, such that it does not send too fast to cause congestion collapse [16] but also, to not let other TCP flows have significant effect on their operation (TCP friendliness). The core idea, of altering TCP's sending dynamics, is not new. Building on work by Mathis et al. [22] and Padhye et al. [27], there has been a significant number of proposals on TCP friendliness and rate control [7, 14, 29], including for multimedia [9, 10]. These proposals show that it is possible to alter the transport's sending dynamics while remaining friendly to other Internet traffic.

While the transport layer is ever evolving, adaptation algorithms in the application layer have so far attempted to mask the transport behaviour. As such, three main concepts for adaptation algorithms have been proposed: throughput-based [17, 35], buffer-based [15, 33], and hybrid [32, 36]. Rate-based solutions [20, 21] have also been proposed, but these are yet to be accepted by the DASH industry forum [2].

Work on adaptation algorithms has slowed, with most recent proposals optimising for specific use cases (e.g., [18]). Partly, this is because of the diminishing returns obtained by increasingly complex algorithms [37]. In this work, we identified cases (e.g., multiple clients competing on a constrained link) where the current state-of-the art algorithms perform poorly. We showed that transport changes, e.g., enabling New CWV, can have positive QoE impact, with up to 4% points of improved rebuffering and 12% points of more stable chunk selection. We hope to open a discussion and allow more researchers seeking to improve adaptation algorithms to look into adapting the transport layer to better suit video traffic.

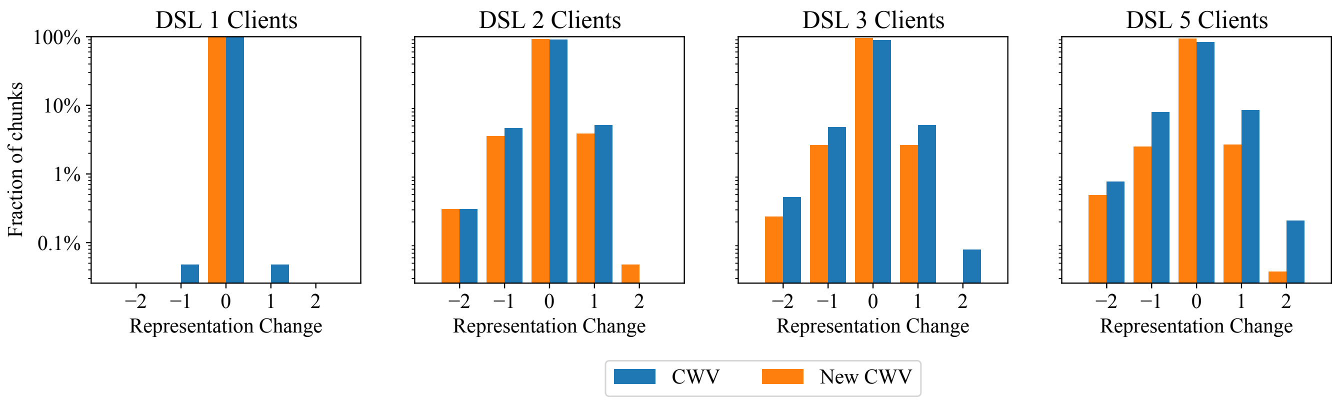


Figure 6: Absolute Bitrate Switches

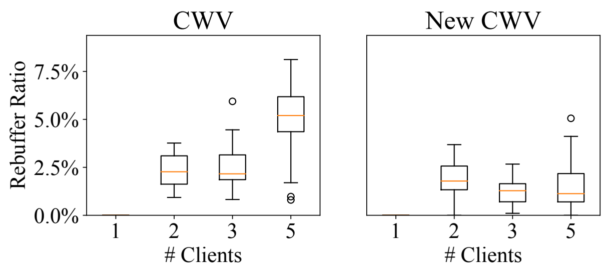


Figure 7: Rebuffer Ratio

We believe that with changes to the transport layer, simple, network-reactive, throughput algorithms will be able to perform comparable to other more complex solutions, such as buffer-based or the dynamic algorithms. In turn, this might enable new work in the field to focus on other aspects improving the adaptation process and not to try and mask the transport's behaviour.

5 CONCLUSIONS

In this paper, we have shown that enabling New CWV improves video playback stability. We compared video delivery with CWV and New CWV, and validated the results shown by previous work [25]. We reported video delivery scenarios using emulated links representative of connections within a country or region, and examined scenarios with different numbers of clients. We found that enabling New CWV, a transport layer change, can improve application layer performance, reducing the number of encoding switches by up to 12% points and rebuffering time by up to 4% points. To sum up, we have shown that transport changes are able to improve the application QoE. We have also shown that these transport changes make the transmission more predictable. We believe, that these improvements would allow simple throughput algorithms to compete with the current state of the art that attempts to mask the transport behaviour. In turn, this might remove the need for adaptation algorithm implementers to mask the transport behaviour, and could instead allow them to focus on other aspects of the adaptation process.

Future work might look at the performance of these algorithms under more dynamic environments, for example, if all clients join the session at random times or in the presence of other cross-traffic.

REFERENCES

[1] [n. d.]. Big Buck Bunny Download Page. https://download.blender.org/demo/movies/BBB/. ([n. d.]).
[2] [n. d.]. DASH Industry Forum | Catalyzing the adoption of MPEG-DASH. https://dashif.org/. ([n. d.]).
[3] [n. d.]. GitHub - rsecchi/newcwv: new-CWV kernel module (draft-ietf-tcpc-newcwv). https://github.com/rsecchi/newcwv. ([n. d.]).
[4] [n. d.]. Recommended upload encoding settings - YouTube Help. https://support.google.com/youtube/answer/1722171. ([n. d.]). (Accessed on 03/09/2022).
[5] [n. d.]. UK Home Broadband Performance: technical report. https://www.ofcom.org.uk/__data/assets/pdf_file/0038/194897/uk-home-broadband-performance.pdf. ([n. d.]).
[6] Saamer Akhshabi, Lakshmi Anantakrishnan, Constantine Dovrolis, and Ali C. Begen. 2012. What happens when HTTP adaptive streaming players compete for bandwidth? *Proceedings of the International Workshop on Network and Operating System Support for Digital Audio and Video* (2012), 9–14. https://doi.org/10.1145/2229087.2229092
[7] Venkat Arun and Hari Balakrishnan. 2018. Copa: Practical Delay-Based Congestion Control for the Internet Copa: Practical Delay-Based Congestion Control for the Internet. In *Proceedings of the 15th USENIX Symposium on Networked Systems Design and Implementation, NSDI 2018*. 329–342. https://www.usenix.org/conference/nsdi18/presentation/arun
[8] Ethan Blanton, Dr. Vern Paxson, and Mark Allman. 2009. TCP Congestion Control. RFC 5681. (Sept. 2009). https://doi.org/10.17487/RFC5681
[9] Gaetano Carlucci, Luca De Cicco, Stefan Holmer, and Saverio Mascolo. 2016. Analysis and design of the google congestion control for web real-time communication (WebRTC). In *Proceedings of the 7th International*

Conference on Multimedia Systems, MMSys 2016. Association for Computing Machinery, Inc, New York, NY, USA, 133–144. https://doi.org/10.1145/2910017.2910605

[10] Soo Hyun Choi and Mark Handley. 2007. Fairer TCP-friendly congestion control protocol for multimedia streaming applications. In *Proceedings of 2007 ACM CoNEXT Conference - 3rd International Conference on Emerging Networking EXperiments and Technologies, CoNEXT*. ACM Press, 1. https://doi.org/10.1145/1364654.1364717

[11] Dash-Industy-Forum. [n. d.]. Dash-Industry-Forum/dash.js: A reference client implementation for the playback of MPEG DASH via Javascript and compliant browsers. https://github.com/Dash-Industry-Forum/dash.js. ([n. d.]).

[12] Jairo Esteban, Steven A. Benno, Andre Beck, Yang Guo, Volker Hilt, and Ivica Rimac. 2012. Interactions between HTTP adaptive streaming and TCP. *Proceedings of the International Workshop on Network and Operating System Support for Digital Audio and Video* (2012), 21–26. https://doi.org/10.1145/2229087.2229094

[13] Gorry Fairhurst, Arjuna Sathiaseelan, and Raffaello Secchi. 2015. Updating TCP to Support Rate-Limited Traffic. RFC 7661. (Oct. 2015). https://doi.org/10.17487/RFC7661

[14] Mark J. Handley, Jitendra Padhye, Sally Floyd, and Joerg Widmer. 2008. TCP Friendly Rate Control (TFRC): Protocol Specification. RFC 5348. (Sept. 2008). https://doi.org/10.17487/RFC5348

[15] Te Yuan Huang, Ramesh Johari, Nick McKeown, Matthew Trunnell, and Mark Watson. 2015. A buffer-based approach to rate adaptation: Evidence from a large video streaming service. In *Computer Communication Review*, Vol. 44. Association for Computing Machinery, 187–198. https://doi.org/10.1145/2619239.2626296

[16] V. Jacobson. 1988. Congestion avoidance and control. *ACM SIGCOMM Computer Communication Review* 18, 4 (aug 1988), 314–329. https://doi.org/10.1145/52325.52356

[17] Junchen Jiang, Vyas Sekar, and Hui Zhang. 2014. Improving fairness, efficiency, and stability in HTTP-based adaptive video streaming with festive. *IEEE/ACM Transactions on Networking* 22, 1 (feb 2014), 326–340. https://doi.org/10.1109/TNET.2013.2291681

[18] Theo Karagkioules, Rufael Mekuria, Dirk Griffioen, and Arjen Wagenaar. 2020. Online learning for low-latency adaptive streaming. *Proceedings of the 11th ACM Multimedia Systems Conference* (2020). https://doi.org/10.1145/3339825

[19] Jonathan Kua, Grenville Armitage, and Philip Branch. 2017. A Survey of Rate Adaptation Techniques for Dynamic Adaptive Streaming over HTTP. (jul 2017), 1842–1866 pages. https://doi.org/10.1109/COMST.2017.2685630

[20] Zhi Li, Xiaoqing Zhu, Joshua Gahm, Rong Pan, Hao Hu, Ali C. Begen, and David Oran. 2014. Probe and adapt: Rate adaptation for HTTP video streaming at scale. *IEEE Journal on Selected Areas in Communications* 32, 4 (2014), 719–733. https://doi.org/10.1109/JSAC.2014.140405 arXiv:1305.0510

[21] Chenghao Liu, Imed Bouazizi, and Moncef Gabbouj. 2011. Rate adaptation for adaptive HTTP streaming. *MMSys'11 - Proceedings of the 2011 ACM Multimedia Systems Conference* (2011), 169–174. https://doi.org/10.1145/1943552.1943575

[22] Matthew Mathis, Jeffrey Semke, Jamshid Mahdavi, and Teunis Ott. 1997. The macroscopic behavior of the TCP congestion avoidance algorithm. *ACM SIGCOMM Computer Communication Review* 27, 3 (jul 1997), 67–82. https://doi.org/10.1145/263932.264023

[23] Ayush Mishra, Xiangpeng Sun, Atishya Jain, Sameer Pande, Raj Joshi, and Ben Leong. 2019. The Great Internet TCP Congestion Control Census. *Proceedings of the ACM on Measurement and Analysis of Computing Systems* 3, 3 (dec 2019), 1–24. https://doi.org/10.1145/3366693

[24] Ricky K.P. Mok, Xiapu Luo, Edmond W.W. Chan, and Rocky K.C. Chang. 2012. QDASH: A QoE-aware DASH system. In *MMSys'12 - Proceedings of the 3rd Multimedia Systems Conference*. ACM Press, 11–22. https://doi.org/10.1145/2155555.2155558

[25] Sajid Nazir, Ziaul Hossain, Raffaello Secchi, Matthew Broadbent, Andreas Petlund, and Gorry Fairhurst. 2014. Performance Evaluation of Congestion Window Validation for DASH Transport. In *Proceedings of Network and Operating System Support on Digital Audio and Video Workshop*. Association for Computing Machinery (ACM), 67–72. https://doi.org/10.1145/2597176.2578275

[26] Chris Newman and Graham Klyne. 2002. Date and Time on the Internet: Timestamps. RFC 3339. (July 2002). https://doi.org/10.17487/RFC3339

[27] Jitendra Padhye, Victor Firoiu, Don Towsley, and Jim Kurose. 1998. Modeling TCP Throughput: A Simple Model and its Empirical Validation *. *Proceedings of the ACM SIGCOMM '98 conference on Applications, technologies, architectures, and protocols for computer communication - SIGCOMM '98* (1998). https://doi.org/10.1145/285237

[28] Jitendra Padhye, Sally Floyd, and Mark J. Handley. 2000. TCP Congestion Window Validation. RFC 2861. (June 2000). https://doi.org/10.17487/RFC2861

[29] Dario Rossi, Claudio Testa, Silvio Valenti, and Luca Muscariello. 2010. LEDBAT: The new BitTorrent congestion control protocol. *Proceedings - International Conference on Computer Communications and Networks, ICCCN* (2010). https://doi.org/10.1109/ICCCN.2010.5560080

[30] Sandvine. 2019. The Global Internet Phenomena Report. (2019).

[31] Michael Seufert, Sebastian Egger, Martin Slanina, Thomas Zinner, Tobias Hoßfeld, and Phuoc Tran-Gia. 2015. A Survey on Quality of Experience of HTTP Adaptive Streaming. *IEEE Communications Surveys and Tutorials* 17, 1 (jan 2015), 469–492. https://doi.org/10.1109/COMST.2014.2360940

[32] Kevin Spiteri, Ramesh Sitaraman, and Daniel Sparacio. 2019. From theory to practice: Improving Bitrate adaptation in the DASH reference player. *ACM Transactions on Multimedia Computing, Communications and Applications* 15, 2s (aug 2019). https://doi.org/10.1145/3336497

[33] Kevin Spiteri, Rahul Urgaonkar, and Ramesh K. Sitaraman. 2016. BOLA: Near-optimal bitrate adaptation for online videos. In *Proceedings - IEEE INFOCOM*, Vol. 2016-July. Institute of Electrical and Electronics Engineers Inc. https://doi.org/10.1109/INFOCOM.2016.7524428

[34] Denny Stohr, Alexander Frömmgen, Amr Rizk, Michael Zink, Ralf Steinmetz, and Wolfgang Effelsberg. 2017. Where are the sweet spots? A systematic approach to reproducible DASH player comparisons. In *MM 2017 - Proceedings of the 2017 ACM Multimedia Conference*. Association for Computing Machinery, Inc, New York, NY, USA, 1113–1121. https://doi.org/10.1145/3123266.3123426

[35] Yi Sun, Xiaoqi Yin, Junchen Jiang, Vyas Sekar, Fuyuan Lin, Nanshu Wang, Tao Liu, and Bruno Sinopoli. 2016. CS2P: Improving video bitrate selection and adaptation with data-driven throughput prediction. In *SIGCOMM 2016 - Proceedings of the 2016 ACM Conference on Special Interest Group on Data Communication*. Association for Computing Machinery, Inc, New York, NY, USA, 272–285. https://doi.org/10.1145/2934872.2934898

[36] Cong Wang, Amr Rizk, and Michael Zink. [n. d.]. SQUAD: A Spectrum-based Quality Adaptation for Dynamic Adaptive Streaming over HTTP. *Proceedings of the 7th International Conference on Multimedia Systems* ([n. d.]). https://doi.org/10.1145/2910017

[37] Xiaoqi Yin, Abhishek Jindal, Vyas Sekar, and Bruno Sinopoli. 2015. A Control-Theoretic Approach for Dynamic Adaptive Video Streaming over HTTP. *Computer Communication Review* 45, 4 (aug 2015), 325–338. https://doi.org/10.1145/2785956.2787486