Assessing the Impact of QUIC on Network Fairness

Romuald Corbel[1], Stéphane Tuffin[1], Annie Gravey[2], Xavier Marjou[1] and Arnaud Braud[1]
[1] Orange Labs, France
[2] IMT Atlantique , France
Email: {romuald.corbel, stephane.tuffin, xavier.marjou, arnaud.braud}@orange.com1, annie.gravey@imt-atlantique.fr

Abstract —As user applications require always more throughput and less latency, new transport protocols such as Quick UDP Internet Connections (QUIC) are proposed. QUIC is currently deployed by Google and is targeted to replace both Transmission Control Protocol (TCP) and Transport Layer Security (TLS) for HyperText Transfer Protocol (HTTP) transport. The flexibility of QUIC, which is implemented in user-space, can however, notably influence the fairness in the sharing of network resources. In this work, we are interested in evaluating the behavior of TCP and QUIC when they coexist together in the network with the same Congestion Control Algorithm (CCA), namely CUBIC. We concretely quantify session level fairness between QUIC and TCP on a platform that injects real traffic over emulated network conditions. Results show that a configurable parameter in QUIC stacks: the number of TCP connections emulated by a single QUIC connection has a strong impact on fairness.

Index Terms—QUIC, TCP, Congestion Control, CUBIC, fairness.

I. Introduction

The Internet protocol stack has continually evolved over time in order to improve the performance of connection- oriented web applications, notably in terms of reliability, latency and delivered throughput. The evolution of users' requirements (regarding e.g., HD/UHD video streaming, virtual reality, etc..) as well as global traffic increase have led to the design of new transport protocols such as Quick UDP Internet Connections (QUIC) [1], which is intended to replace both Transmission Control Protocol (TCP) and Transport Layer Security (TLS) for HyperText Transfer Protocol (HTTP) transport. QUIC is already widely deployed on Internet, particularly by Google, and currently represents about 7% of global Internet traffic [2]. Transport protocols such as QUIC and TCP influence the performance of web services as they include Congestion Control Algorithms (CCAs) running on network endpoints (e.g. smartphones, PCs, servers). Indeed, CCAs are intended to adapt application data rate in order to avoid congestion, but also to efficiently use available bandwidth. The co-existence of several transport protocols raises the "fairness" issue: when several data flows share a bottleneck, is the available bandwidth "fairly" shared between them? Network operators are obviously concerned with fairness, as they wish to satisfy all their customers by providing them with a fair share of available resources according to their needs.

Many versions of TCP have co-existed in the past. The new issue raised by the co-existence of legacy and new transport protocols, such as QUIC, is the implementation of CCA within the user-space and at the application-layer of both endpoints (clients and servers), whereas legacy TCP transport protocols implement the CCA within the kernel-space of the Operating System (OS). Deploying transport protocol modifications is thus easier for QUIC than for TCP. When it comes to the CCA, this potentially influences network performances and notably network fairness in case of congestion.

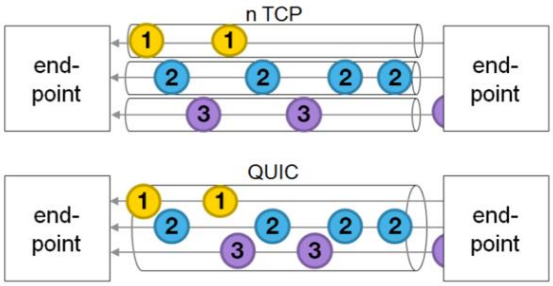


Fig. 1. 3 TCP flows versus a single QUIC flow with 3 streams

As illustrated in Fig. 1, while TCP-based transport relies on multiple connections (i.e. flows) for transmitting multiple resources bundled in HTTP (e.g., video, images, text), QUIC transports multiple streams within a single data User Datagram Protocol (UDP) flow.

The present paper analyses the competition between TCP and QUIC flows while varying the number of emulated TCP connections within QUIC in order to assess the impact of this number on network fairness. More precisely, network fairness is quantified under the assumption that QUIC and TCP compete in a single bottleneck while using the same loss-based CCA, namely CUBIC. Session level fairness is assessed based on Jain's fairness index [3]. We use a test-bed described in [4] for injecting real TCP and QUIC traffic under various emulated network conditions.

The rest of the paper is organized as follows: Section II provides background information on QUIC and CUBIC while section III focuses on how multiple TCP connections are emulated by QUIC. Section IV specifies how session fairness is measured. Fairness assessment results are presented in Section V. Related work is discussed in Section VI. Finally, our main conclusions are exposed in Section VII.

Manuscript received March 25, 2019; revised September 2, 2019.
doi:10.12720/jcm.14.10.908-914

II. Background on QUIC and CUBIC

In the last few years, many research efforts have been directed at the evolution of the IP protocol stack. In 2012, QUIC was proposed as a new transport protocol in order to improve the perceived performance of connection-oriented web applications. QUIC is currently being studied at the Internet Engineering Task Force (IETF) and is targeted to replace TCP, at least partially.

A. QUIC is a Nutshell

QUIC targets transport performance improvement by *(i)* reducing handshake delay (QUIC allows to exchange data from the first packet while TCP and TLS require several Round-Trip Times (RTT) to establish the connection) and *(ii)* avoiding Head-of-line (HOL) blocking (QUIC multiplexes various connections between two endpoints over the same UDP flow). In themselves, these features have no direct impact on network fairness; however, as always, the devil is in the details since other QUIC features, notably pacing and packet recovery are related to stream multiplexing and do potentially impact network fairness.

B. Evolution of Congestion Control Algorithms

New CCA for TCP have been proposed, e.g. CUBIC [5] in 2005 and BBR [6] in 2015. Compared to legacy CCAs (notably RENO), both aim at improving performances by providing data rate stability and better fairness.

In order to increase network throughput, CUBIC modifies the linear growth of the congestion window size used during the congestion avoidance phase in e.g. RENO by a so-called "cubic" growth characterized by the cubic function:

$$W(t) = C(t-K)^3 + W_{max} \quad (1)$$

In congestion avoidance (i.e., after slow-start or after a congestion event) CUBIC quickly increases the window size until a "saturation point" (W_{max}) shown in Fig. 2. After the saturation point, the window increase is slower. The first concave, and then convex, behavior of the window size growth is intended to improve performance as the window size stays close to a (locally) optimal W_{max} value.

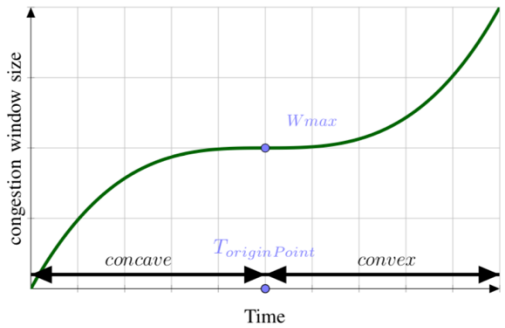


Fig. 2. Cubic growth of the congestion window.

III. QUIC Emulation of Multiple TCP Connections

CUBIC is widely used by TCP as CCA. It is also implemented in new transport protocols such as QUIC. In particular, QUIC introduces additional parameters in order to emulate the behavior of several TCP connections that are replaced by the single UDP flow multiplexing the data streams which would otherwise be transported in distinct TCP connections.

The emulation of multiple TCP connections by QUIC is implemented in various QUIC stacks (notably QUIC-GO [7], proto-QUIC [8] and Chromium [9] among others) although this feature is not documented by the IETF in RFC 8312 [10].

The number N of emulated TCP connections impacts the behavior of the congestion window which in turn determines the allowed data rate (i.e. the number of bytes that can be sent without acknowledgement). The present study focuses on QUIC-GO which allows configuring N, as a constant, in it's source code.

A. Window Size Decrease

The congestion window is decremented after a congestion event (i.e. packet loss detection or timeout). The congestion windows size cWS_d is decreased as follows:

$$cWS_d = B * cWS_{b-c} \quad (2)$$

where cWS_{b-c} is the window size before the congestion event, $B = (N - 1 + \beta)/N$ and $\beta = 0.7$. B is thus an increasing function of N, the number of emulated TCP connections. This implies that a congestion event yields a smaller decrease in the allowed data rate when N is large than when it is small.

This is illustrated in Fig. 3a that shows how the window size is modified by a congestion event for N in $\{1, 2, 5\}$. Assuming the congestion event occurs just before $350ms$, the congestion windows size (i.e. the allowed data rate) decreases differently, depending on N. Indeed, we observe that, when the congestion control algorithm emulates a single TCP connection, the allowed data rate is reduced by 30% whereas it is reduced by barely 3% when the CCA emulates 10 TCP connections.

B. Window Size Increase

In general, increasing the congestion window size is done in two phases, slow-start and congestion avoidance.

In the slow-start phase, the windows size increase is ex- ponential as shown in eq. (3), which controls the congestion window increase and is applied at each acknowledged segment.

$$cWS_d = cWS + MSS \quad (3)$$

where cWS is the congestion windows size and MSS is the segment size. Clearly, in this phase, the congestion window increase does not depend on the number of emulated TCP connections N.

In the following, we thus focus on the congestion avoidance phase. In the congestion avoidance phase the window size increase is cubic (for CUBIC) or linear (for RENO). The allowed data rate increase is derived according to four steps, two of those being impacted by the value of N.

(i) Congestion window increase is controlled by the cubic function eq. (1). CUBIC computes the difference Δ_{cWS} between the cubic function and W_{max}(plateau of the cubic function). Generally, W_{max} is the window size value just before the congestion event. Hence,

$$\Delta_{cWS} = CCWS * \text{offset}^3 * MSS \gg CS \quad (4)$$

where $CCWS$ is the Cube Congestion Window Scale, CS is the Cube Scale, MSS is segment size. The variable offset is given by $\text{offset} = f(T_{originPoint}, T_{ACK})$. $T_{originPoint}$ corresponds to the time at which the cubic function is equal to W_{max} and T_{ACK} represents the time when the packet is acknowledged. $T_{originPoint}$ is calculated after each congestion event and can take two values depending on network conditions: 0 or $\sqrt[3]{CF * (lastCWS_{max} - cWS_d)}$ where CF is the Cube Factor, $lastCWS_{max}$ is the previous maximum value of the window size and cWS_d is the windows size occurs after a congestion event (see eq. (2)). Thus, the value for $T_{originPoint}$ depends on the number of emulated TCP connections N. The larger N the quicker the window size reaches W_{max}. The QUIC-GO implementation sets the following default values: $CCWS = 410$, $CF = 1 << CS/CCWS/MSS$, $CS = 40$, and $MSS = 1460$.

(ii) Defining m as the number of acknowledged bytes, CUBIC limits the window size increase to $cWS_{threshold}$:

$$cWS_{threshold} = cWS_{t-1} + \frac{m}{2} \quad (5)$$

(iii) To ensure that the window size achieved by CUBIC is at least equal to the one that RENO could achieve, CUBIC sets the new window size as the maximum between the current cWS (CUBIC) and the estimated RENO congestion window size $eCWS$ of RENO. The estimated window size $eCWS$ is given by:

$$eCWS = eCWS_{t-1} + \frac{m * \propto * MSS}{eCWS_{t-1}} \quad (6)$$

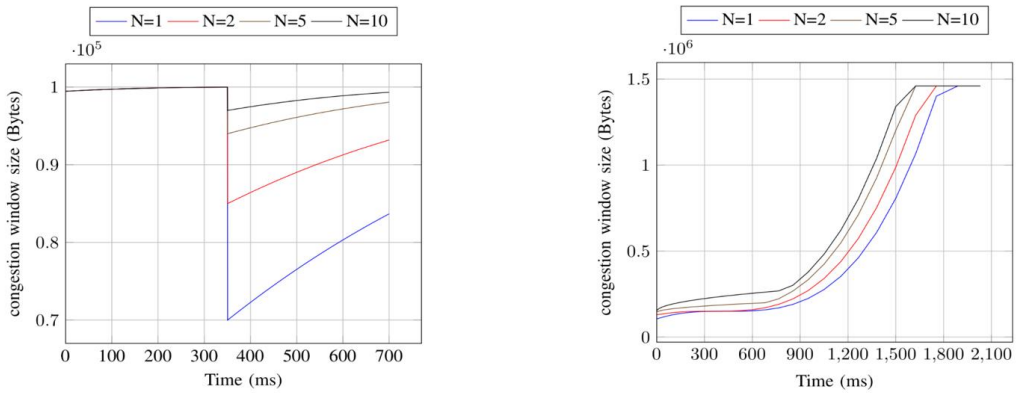
where α is defined by $\alpha = 3 * N^2 * (1 - B)/(1 + B)$. The above formula shows that $eCWS$ depends on N, and implies that the data rate increases faster for higher values of N.

(iv) Lastly, CUBIC limits the maximum window size to cWS_{max}:

$$cWS_{max} = MAX_p * MSS \quad (7)$$

where MAX_p is the maximum number of packets in the congestion window. In QUIC-GO MAX_p is set to 1000.

The window size evolution during congestion avoidance is illustrated in Fig. 3b for various different values of N. A congestion event occurs just before $0ms$. We assume that the previous congestion window is equal to 150,000 bytes and that W_{max} is 100,000 bytes. We also assume that an ACK is produced every segment and that RTT equals $50ms$. Fig. 3b presents three stages: during the first $800ms$, the window size increases according to the cubic function for $N = 1$ and $N = 2$. In the same period of time, for larger values of N, the RENO window size increase exceeds the CUBIC one; the resulting congestion window size thus presents a linear increase according to step *(iii)*. Between $800ms$ and $1500ms$ the window size increase is set to $cWS_{\text{threshold}}$ according to step *(ii)*, irrespective of N: this corresponds to a linear increase with the same slope for all N. Finally, after *1500ms*, the window size is set to the maximum value cWS_{max} specified by CUBIC. Obviously, the larger is N, the quicker the window size increases.



(a) decreasing window size (b) increase window size

Fig. 3. Evolution of the congestion window size according to the number of emulated TCP connection.

IV. Session Fairness Assesment

In principle, network fairness consists in allocating network resources while maximizing users' satisfaction according to their needs. In transport protocols, fairness is generally thought in terms of network throughput. In this section we present the network fairness metric that will be used to quantify the equity among concurrent data flows in a bottleneck.

Several fairness metrics have been proposed in the literature, the most well-known and used metric being the Jain's index [3]; assuming that flow i receives xi at a given bottleneck, the Jain's fairness index is computed as:

$$J_{x_1,x_2,\ldots,x_n} = \frac{\left(\sum_{k=1}^{n} x_k\right)^2}{n\sum_{k=1}^{n} x_k^2} \quad (8)$$

where xi is the ratio between the observed throughput and the "fair" throughput according to max-min fairness (this is because some flows may be regulated by another bottleneck). Jain's index value ranges between $1/n$ (worst case) to 1 (best case), and is maximum when all flows receive the same allocation.

Network operators are interested in evaluating the fairness during the whole period of flow competition. In this context, a simple way of determining the fairness over a period of time is to derive a global "session fairness" indicator from successive values of the Jain's index computed over small intervals of time. Computing the mean of the successive Jain's index is not indicative of session fairness, as what happens in successive intervals may even out local unfairness. Taking this into account, when assessing session fairness between two flows:

$(n = 2)$ which share a given bottleneck, it is useful to define the "dominant" flow at each instantaneous measure as the flow which obtains a larger throughput than the other during the considered time interval. We thus split the competition time between two flows A and B into S slots of 100 milliseconds. For each time slot i, the fairness index is given by j_i and dominance between A and B during slot i is characterized by di where

$$d_i = \begin{cases} -1 & if\ A\ is\ dominant \\ 1 & if\ B\ is\ dominant \end{cases} \quad (9)$$

Then, a global session fairness metric F is defined by considering both $J = \{j1, \ldots, jS\}$ and $d = \{d1, \ldots, dS\}$ as follows:

$$F = \frac{\sum_{i=1}^{S} d_i * (1 - j_i)}{S} \quad (10)$$

F takes its values in $[-0.5, +0.5]$. The sign for F deter- mines the "globally dominant" flow. When B (respectively A) is globally dominant, F is positive (respectively negative). The F value also indicates a session level fairness, i.e., when $|F|$ is close to 0, session fairness is achieved whereas a global unfairness is characterized by larger values for |F|.

V. Performance Analysis of QUIC and TCP

In the following, we use the session fairness indicator defined in Section IV to assess the impact of N on the fairness achieved when flows transported over QUIC and TCP share a single bottleneck. We assume that both QUIC and TCP use CUBIC as CCA.

A. Test-bed Description

QUIC and TCP-based services are implemented in the test-bed illustrated in Fig. 4. We use the TCP stack provided by the Linux kernel packaged in Ubuntu 16-04 and the QUIC-GO [7] implementation of the QUIC protocol specified by the IETF in [11]. In our setting, both QUIC and TCP use CUBIC as CCA. The Linux kernel parameter *tcp_no_metrics* save is enabled in order to not reuse previous metrics as initial conditions for new TCP connections. This would otherwise introduce a bias in our measurements since QUIC-GO has no equivalent feature and since network conditions are changed between test occurrences. The TCP optimization parameters TCP Segmentation Offload (TSO) and Hystart are also dis- abled for a fair comparison as there is no equivalent in QUIC-GO. A file transfer service uses either QUIC or TCP to download a $10MB$ file.

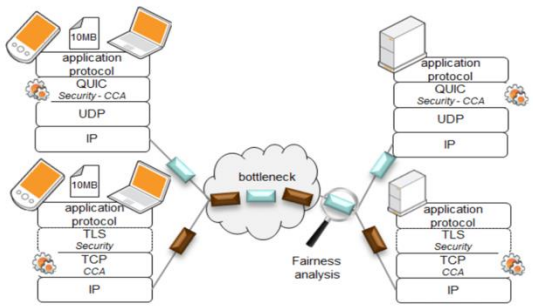


Fig. 4. Test-bed: competition between QUIC and TCP.

The emulated network conditions have been obtained by varying four network characteristics: latency, residual loss rate (for losses that are not due to queuing), buffer size and queuing policies. The link bandwidth (bottleneck) is set to $1Mb/s$. The used network features are given in Table I.

TABLE I: NETWORK AND SESSION FEATURES

Latency (ms)	50, 300
Loss rate (%)	0, 0.1, 0.5, 1, 2, 5
Buffer size (bytes)	11250, 62250
N (emulated TCP connection)	1, 2, 5

We computed the global session fairness metric for the 96 scenarios described in Table I; each scenario corresponds to a different combination of network level or session level characteristics.

B. Fairness Analysis

Experiments first showed that neither network latency nor buffer size have an impact on session fairness; when varying both latency and buffer size the obtained fairness index F is almost constant (with a variance of 10^{-4}). This is not illustrated here due to lack of space.

We then set latency to $300ms$ and the buffer size to 62250 bytes (the double of the Bandwidth-Delay Product (BDP)), and consider different values for loss rate and N.

The session fairness measurement results are illustrated in Figs. 5 and 6. Ten independent tests (file

transfer) are represented in the x-axis while different loss rates are represented in the y-axis. Each file transfer result is characterized by two indicators: global fairness value and dominant flow. The fairness value is illustrated by a map of colors where green and red respectively represent fairness and unfairness between both data flows. The dominant flow is indicated by a geometrical shape: a blue circle (respectively a red square) corresponds to QUIC (respectively TCP) being globally dominant.

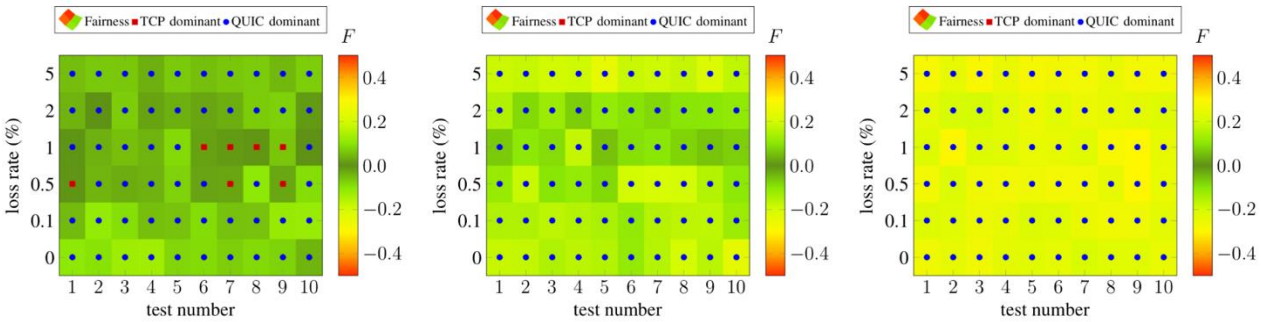


Fig. 5. Fairness evaluation of a single TCP connection competing with a single QUIC connection.

To analyze the impact of fairness according to the number of emulated TCP connections in QUIC, we consider the two following cases: (i) a single TCP connection competing with a single QUIC connection emulating N TCP connections, (ii) N TCP connections competing with a single QUIC connection emulating N TCP connections.

1) a single TCP connection competing with a single QUIC connection emulating N TCP connections: Fig. 5 illustrates the fairness of throughput sharing between a single TCP connection and a single QUIC connection.

The first observation to make is that the packet loss value has no visible impact on the obtained results. On the other hand, it is seen that $|F|$ increases (which means more unfairness) when N increases. The average value of the fairness index is equal to 0.05 (fair) when $N = 1$ and 0.248 (partially fair) when $N = 5$. As shown in Section III, this is due to how CUBIC is implemented in QUIC: during the congestion avoidance, for large N, QUIC increases the data rate faster than TCP, likewise QUIC decreases the data rate more slowly than TCP after a congestion event.

These experiments show that N impacts on network fairness: depending on N, bottleneck sharing between TCP and QUIC can be globally fair (small N) or not (large N).

2) N TCP connections competing with a single QUIC connection emulating N TCP connections: Fig. 6 illustrates the fairness results obtained when N TCP connections compete with a single QUIC connection emulating N TCP connections, for various N. Once again, it is seen that the packet loss value has no impact on the obtained results. Results show network fairness is globally achieved in all considered cases. The average value of F is always between -0.1 and 0.1, which corresponds to a fair sharing. However, even if resource sharing is globally fair, the variability in the dominant flow is evidenced when considering $N = 1$ and $N = 5$ (compare e.g. Figs. 6a and 6c). While QUIC is dominant for $N = 1$, TCP is dominant when $N = 5$. This phenomenon can be explained by the limitation of the windows size implemented in QUIC's CUBIC (see eq. (7)). Thus, while QUIC is not able to increase the window size when the maximum is reached, each TCP connection can independently augment its own window size.

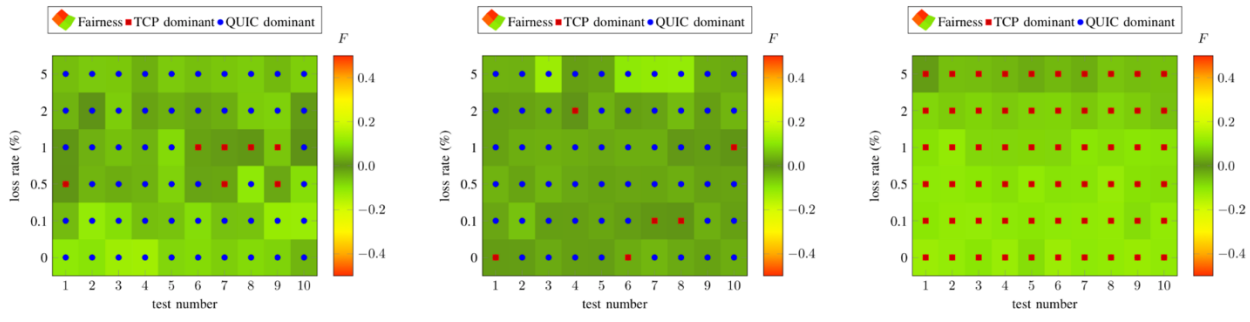


Fig. 6. Fairness evaluation of various TCP connections competing with a single QUIC connection.

VI. Related Work

QUIC was introduced in IETF in 2016 [12] and is already deployed on the Internet (for example when Google Chrome is used to access Google services). Since then, various studies about QUIC performances and its impact on fairness have been published. An evaluation of QUIC performances, reported in [13], demonstrates that QUIC is faster, according to a page loading time metric, than TCP in the context of web browsing. However, the authors do not analyze the fairness of QUIC regarding other concurrent flows. In [14], the authors propose an analysis of the fairness between QUIC and TCP. They report that QUIC is globally unfair with TCP as QUIC obtains a larger data rate than TCP, irrespective of the value of N, and of the number of TCP connections competing with a QUIC flow. This drastically differs from our results. This probably lies in the considered QUIC "calibration" considered in [14]. Indeed, the authors calibrate the QUIC stack to make it behave similarly to the QUIC servers deployed by Google. Additionally, they used the so-called Google-QUIC (gQUIC) protocol that preceded the current draft IETF specification implemented by the QUIC stack used for our measurements.

Regarding the evaluation of fairness when a single con- nection emulating several TCP connections competes with several independent TCP connections, different studies are also present in the literature. In particular, we note the study reported in [15]. The authors analyze the relative data rate of one MulTCP(N) connection emulating N TCP connections against the data rate of a single TCP connection. They show that for small N, the relative share of bandwidth of MulTCP is N times the single TCP connections share. However as N increases, the relative share of bandwidth of MulTCP stops increasing. This study is based on the analysis of four congestion control algorithms (TCP Tahoe, TCP Reno, New Reno and TCP Sack). The emulation of N TCP connections with CUBIC, on which the present paper focuses, has not been studied to the best of our knowledge.

VII. Conclusions

New transport protocols such as QUIC are receiving growing attention from researchers. QUIC is intended to improve the QoE delivered to connection-oriented web applications by solving problems such as HOL blocking.

However, the QoE improvement of certain web applications should not be obtained by degrading network fairness, which is a growing concern for network operators.

In the work reported in the present paper, we analyzed the impact of the coexistence of new and legacy transport protocols, i.e. QUIC and TCP, which both use the same loss-based CCA (CUBIC). We implemented a test bed in order to assess the competition between TCP and QUIC over a single fixed bottleneck while emulating real network conditions according to latency, packet loss and buffer size.

The obtained results show that network conditions (latency, packet loss, buffer size) do not have a significant impact on fairness. On the other hand, the number N of emulated TCP connections by QUIC's CUBIC has a strong influence on fairness. Indeed, QUIC is shown to monopolize N times more bandwidth than a single TCP connection, and thus to behave unfairly compared to standard TCP connections. When we compare the rate achieved by a single QUIC connection emulating N TCP connections, we however notice that the global throughput reached by N independent TCP connections is slightly larger than the one reached by the single QUIC connection although global fairness is achieved.

The open problem revealed by the present study is the unfairness related to the value of N. Section I points out that, whereas modifications to TCP implementations typically imply an OS upgrade, implementing QUIC with a large N value can be easily done in the application-space, independently of the network and of the OS. This should motivate the introduction of either engineering rules or standards specifying how to manage the number of emulated TCP connections in order to guarantee a fair sharing of bottleneck bandwidth between QUIC and TCP.

Acknolegdement

We thank Isabelle Hamchaoui and Alexandre Ferrieux for their precious support and insightful assistance. Their valuable help enabled us to greatly improve this paper.

References

[1] J. Iyengar and M. Thomson, "QUIC: A UDP-Based multiplexed and secure transport," IETF, Internet-Draft draft-ietf-quic-transport-17, 2018.

[2] A. Langley, *et al*., "The QUIC transport protocol: Design and internet-scale deployment," in *Proc. Conference of the ACM Special Interest Group on Data Communication*, ser. SIGCOMM '17. New York, NY, USA: ACM, 2017, pp. 183–196.

[3] R. K. Jain, D. M. W. Chiu, and W. R. Hawe, "A quantitative measure of fairness and discrimination for resource allocation in shared computer systems," DEC-TR-301, Digital Equipment Corporation, Tech. Rep., Sep. 1984.

[4] R. Corbel, A. Braud, X. Marjou, G. Simon, and A. Gravey, "Fair-ness measurement procedure for the evaluation of congestion control algorithms," in *Proc. 5th International Conference on Communications and Network Engineering*, 2018.

[5] S. Ha, I. Rhee, and L. Xu, "CUBIC: A new TCP-friendly high-speed TCP variant," *SIGOPS Oper. Syst. Rev.*, vol. 42, no. 5, pp. 64–74, Jul. 2008.

[6] N. Cardwell, Y. Cheng, C. S. Gunn, S. H. Yeganeh, and V. Jacobson, "Bbr: Congestion-based congestion control," *Queue*, vol. 14, no. 5, pp. 50:20–50:53, Oct. 2016.
[7] QUIC-GO. [Online]. Available: https://github.com/lucas-clemente/quic- go
[8] proto-QUIC. [Online]. Available: https://github.com/google/proto-quic
[9] chromium. [Online]. Available: https://github.com/chromium/chromium
[10] I. Rhee, L. Xu, S. Ha, A. Zimmermann, L. Eggert, and R. Scheffenegger, "CUBIC for fast long-distance networks," RFC 8312, Feb. 2018.
[11] J. Iyengar and M. Thomson, "QUIC: A UDP-Based multiplexed and secure transport," IETF, Internet-Draft, 2018.
[12] R. Hamilton, J. Iyengar, I. Swett, and A. Wilk, "QUIC: A UDP-Based multiplexed and secure transport," Internet Engineering Task Force, Internet-Draft draft-hamilton-quic-transport-protocol-00, 08 2016.
[13] P. Megyesi, Z. Krmer, and S. Molnr, "How quick is QUIC?" in *Proc. IEEE International Conference on Communications (ICC)*, May 2016, pp. 1–6.
[14] A. M. Kakhki, S. Jero, D. Choffnes, C. Nita-Rotaru, and A. Mislove, "Taking a long look at QUIC: An approach for rigorous evaluation of rapidly evolving transport protocols," in *Proc. Internet Measurement Conference*, ser. IMC '17. New York, NY, USA: ACM, 2017, pp. 290–303.
[15] J. Crowcroft and P. Oechslin, "Differentiated end-to-end internet services using a weighted proportional fair sharing TCP," *CoRR*, vol. cs.NI/9808004, 1998.

Romuald Corbel is currently PhD Candidate at IMT Atlantique-France where he received the Engineering degree in Computer Science, Networking and Telecommunications in 2016. Since 2013, he has been with Orange Labs. His research interests center around the upper layers of the IP protocol stack while mainly considering congestion control algorithms.

Stéphane Tuffin is an R&D engineer at Orange Labs since 20 years recognized as an Orange Expert on Future Networks. He has been deeply involved in the transition from circuit-switched voice to IP and spent several years in preparing and supporting IMS deployments in Orange countries.
Attracted by the fast progress of web based real-time communications, in 2013, he took the lead of a project aiming at bootstrapping company assets in this field. This gave birth to Flexible DataSync, a Cloud solution for businesses.
Since 2017, he is leading a research program aiming at giving visibility and control on the quality experienced by users on Orange networks. The program is tackling topics such as "network monitoring with encrypted traffic"; "network optimizations with new transport and applications"; "QoS models for wireless networks" and "QoE driven autonomic control loops". Besides, Stéphane has interests in exploring new ways of interfacing operator networks with verticals.

Annie Gravey received the French "Agrégation de Mathématiques" in 1978 and a PhD degree in Automatics and Signal Theory from Paris-Sud University (France) in 1981. She is currently Professor in the Computer Engineering Department at TELECOM Bretagne, Brest (France), and has have joined the IRISA in January 2012 in the ADOPNET research team. Before joining TELECOM Bretagne in 2000, she spent almost 20 years in France Telecom Research Lab, in Lannion (France), where she analysed or designed mechanisms for specifying and controlling broadband traffic. She also participated for many years to both ITU and ETSI and contributed to the traffic management and QoS specification of ATM and IP standards. Her research interests include the design and evaluation of traffic engineering methods for broadband networks, and of procedures related to QoS specification and management. In particular, in the last years, she has focused on broadband Access networks (Optical, wired and wireless), on the design of Metropolitan Optical networks with sub-wavelength granularity, on fixed/mobile/Wi-Fi integration, and on controlling the deployment of applications on different networks and network elements. Most of these topics are related to evolutional approaches for next generation Internet. Annie Gravey is member of SEE, IEEE and ACM.

Xavier Marjou is an R&D engineer at Orange Labs for 15 years and he also worked previously at Siemens and Alcatel. He has been deeply involved in voice over IP technologies and also participated to the IETF working groups dealing with real-time communications.
Since 2015, Xavier has special interests in exploring collaboration and fairness topics in the field of telecommunications.

Arnaud Braud received a master's degree in both electronics and computer science from "Ecole Nationale Supérieure Des Sciences Appliquées et de Technologie". He has been an R&D engineer at Orange Labs for 15 years now specializing in web technologies for network operators. His current topics of interests are the internet transport protocol stack and the industrial data space.