Impact of QUIC on fairness in mobile networks

Romuald Corbel*, Stéphane Tuffin*, Annie Gravey§, Arnaud Braud* and Xavier Marjou*
Orange Labs*, IMT Atlantique§
{romuald.corbel,stephane.tuffin,arnaud.braud,xavier.marjou}@orange.com*, annie.gravey@ieee.org§

***Abstract*—Transport protocols continuously evolve in order to meet the needs of users and new services. A recent significant example is Quick UDP Internet Connections (QUIC), an experimental protocol introduced by Google, that aims to replace two widely used transport and security protocols, namely Transmission Control Protocol (TCP) and Transport Layer Security (TLS). QUIC is implemented in user applications (rather than in the operating system's kernel) and is meant to be resistant to ossification and therefore more versatile. Because of its evolvable nature, QUIC introduces new challenges in managing fairness. In this paper, we focus on determining the influence of QUIC on the data rate of mobile users. We specifically analyze protocol fairness when TCP and QUIC flows coexist on the wireless link of a mobile network. During our fairness assessments, we identified the impact of the following QUIC implementation aspects: the emulation of multiple TCP connections, the limitation of the congestion windows size and the usage of the hybrid start (hystart). Results show these mechanisms have a strong influence on fairness when the loss rate is low on the mobile network. Indeed, a wrong setting of the default parameters of these mechanisms or the activation of the hystart option can affect the performance of the transport protocols and therefore also the fairness. Finally, the lack of standardization of the emulation of multiple TCP connection in QUIC makes us question more broadly how QUIC's design philosophy accommodates fairness.**

***Index Terms*—QUIC, TCP, Congestion Control, CUBIC, fairness, mobile network.**

I. Introduction

Mobile networks have evolved over time in order to improve user's quality of experience, notably in terms of latency and delivered network throughput. Forthcoming applications (e.g. 4K video, virtual reality, etc.) require huge data rates. Future generations of mobile networks promise greater data rate (5G shall targets ten times 4G data rates), latency reduction (dividing by four current delays) and consequently Quality of Experience (QoE) improvements [1].

The performance of mobile applications depends not only on the wireless channel quality managed by the access network but also on the data transport handled by the layers above IP and in particular on the transport protocols. These protocols strongly influence the final throughput of user applications.

Over time, the Internet protocol stack has continuously evolved to improve the performance of network services and user applications. Quick UDP Internet Connections (QUIC) [2] was introduced by Google in 2012 to replace both Transmission Control Protocol (TCP) and Transport Layer Security (TLS), notably for web connections. It is currently studied by the Internet Engineering Task Force (IETF) for standardization. The main strengths of QUIC [2], [3] are low connection establishment delays ("0-RTT") and encryption at the transport layer which is in particular intended to fight against protocol ossification caused by intermediaries. QUIC is already used by Google-based services, e.g. when using Google browsers (namely, Chrome) and/or Google content servers (namely, YouTube), representing 7% of the world's Internet traffic [2].

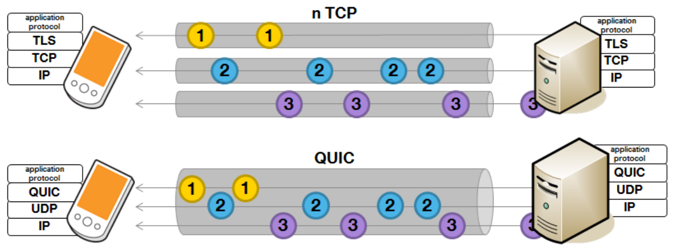


Fig. 1: 3 TCP flows versus a single QUIC flow with 3 streams

The QUIC protocol reuses transport features from the state of the art. Indeed, similarly to the Stream Control Transmission Protocol (SCTP) [4], QUIC is able to multiplex several application sessions (e.g. HyperText Transfer Protocol (HTTP)) on a single transport connection where TCP would instead use several connections to avoid Head-of-line (HOL) blocking; this is illustrated in Fig. 1. In TCP, HOL blocking occurs in case of packet loss when the receiver is waiting for a missing packet to unblock the delivery of previously received packets to the application.

The coexistence of several QUIC- and TCP-based flows raises the classical question of "fairness". This question is even more salient when considering QUIC's versioning abilities to have fast update cycles; this is because QUIC is directly implemented in user applications (e.g. Google Chrome, LiteSpeed Web Server) that are benefiting from faster and simpler modifications [2] compared to TCP which is implemented in the kernel of the Operating System (OS). The number of content providers using personalized transport features and settings can significantly increase. For instance, Google has its own server-side implementation requiring another implementation of QUIC to be calibrated to match its characteristics [5]. Smaller content providers have the same possibility with QUIC but not necessarily the same abilities as Google. Since many QUIC versions can coexist in the network it makes fairness management and control more complex. This hampers network operators efforts in enforcing collective fairness among concurrent flows.

In the present paper, we are notably interested in evaluating the fairness among competing TCP and QUIC flows on a mobile network in order to identify which easily modifiable features in QUIC implementations have a significant impact on fairness. We evaluate various real mobile traffic scenarios in a test-bed which carries both QUIC and TCP flows. We then assess fairness by sampling measurements during the period of competition between the flows. The fairness assessment is then obtained by averaging the sampled values. This Session Fairness Assessment (SFA) method is described in more details in [6].

In particular, we assess the impact of hybrid start (hystart) [7] (embedded in QUIC and TCP) which influences when to switch from the slow start phase to the congestion avoidance phase. In addition, we identify the impact of the number of TCP connections emulated by QUIC and the impact of the limitation of the QUIC congestion window size.

The paper is organized as follows: The related works are discussed in Section II. Section III describes our test-bed whereas Section IV explains how the fairness metric is derived. The impact of several parameters used in QUIC stacks configuration is addressed in Section V: the hystart option is analysed in Section V-A while the number of emulated TCP connections in QUIC is considered in Section V-B; lastly, Section V-C reports the impact of the congestion window size limitation. Section VI discusses the lack of attention given to fairness issues in QUIC design choices. Section VII concludes the paper.

II. Related Work

A. QUIC's fairness in mobile networks

Several studies [5], [6] about QUIC fairness on fixed data rate network have been published. Each QUIC fairness study measured the implementation of multiplexing of several application sessions on the same transport connection.

A common case is the competition between multiple TCP connections with a single QUIC connection transporting multiple sessions, as both methods can be used simultaneously to download web page resources from different domain (supporting different protocol). The practice in web browsers is to open several TCP connections or several QUIC streams in a single QUIC connection for each domain (depending on the protocol implemented on each domain). Congestion control is at the QUIC connection level and not at the stream level. Thus, an application using N TCP connections could theoretically achieve N times the QUIC data rate. In several QUIC implementations (i.e. QUIC-GO [8], Chromium [9] and QUIC Google [10]), the emulation of N simultaneous TCP connections was introduced in the CUBIC congestion control algorithm to make the competition fair. This is done by modifying the evolution of the congestion window size which defines the number of packets that the sender is authorized to send without acknowledgement (the congestion window size is proportional to the data rate of a connection). The emulation of N TCP connections impacts the increase and decrease phases of the congestion window size. In our previous paper [6], we demonstrated that the main impact is on the decrease phase after a packet loss during which the congestion window size is decreased as if only one of the N emulated TCP connection suffered the packet loss.

In [5], the authors present a fairness study on fixed data rate networks. They demonstrate that QUIC is faster than TCP. The used version of QUIC is calibrated to obtain a behavior similar to the servers deployed by Google on the Internet. During their calibration, they show that the initial value of congestion window size impacts the download time of a large file (up to 50%).

Our previous paper [6] shows that, in case of a fixed rate bottleneck, the QUIC protocol corresponding to the IETF draft [3] (without calibration) is fair with TCP when the number of TCP connections emulated by QUIC is the same as the number of competing TCP connections. The present paper proposes a new fairness assessment with QUIC and TCP both using the same congestion control algorithm (CUBIC) and competing on sharing a variable rate bottleneck typical of those observed in mobile networks.

To the best of our knowledge, this study is the first to measure the impact on the fairness of the number of emulated TCP connections in QUIC with a competing TCP flow on a mobile networks. To do this, we do not calibrate the QUIC protocol to match Google's servers, but we take an open-source client and server, as a content provider would do.

B. Impact of hystart on the fairness

When initializing a new transport connection, the transport protocol has no knowledge of the bandwidth available on the network. In order to quickly grab all available resources, the congestion control algorithm starts increasing the window size with a "slow-start" phase. For example, by default in Linux, the congestion window size is initialized to 10 and increases exponentially till it switches to the congestion avoidance phase, during which the pace of congestion window size increase is slower.

Different events can trigger the transition between these two phases:

- *Packet loss event:* When a packet loss event occurs, i.e. the network is congested, then the congestion window size is reduced (e.g. by one third with CUBIC) and the congestion control algorithm switches to the congestion avoidance phase. This reaction to packet loss is implemented in congestion control algorithms such as CUBIC, RENO and Bottleneck Bandwidth and Round-trip propagation time (BBR).
- *Reaching a threshold:* When the congestion window size reaches a certain threshold value, the congestion control algorithm switches to congestion avoidance. The default threshold value in the Linux implementation of CUBIC is 644 packets.
- *Hystart:* Proposed in 2011, the hystart option was presented in these papers [7], [11] and is part of RFC 8312 [12]. hystart is implemented as an option in both TCP and QUIC. The hystart option is presented as a solution to determine a "safe" exit point from the slow-start, i.e. to switch from slow-start to congestion avoidance.

If an increase in Round-Trip Time (RTT) is detected during slow start, hystart considers that this indicates an incipient network congestion and switches from slow start to congestion avoidance in order to avoid packets loss. The hystart option is widely deployed on the Internet. Indeed, the hystart option is enabled by default on Linux and in different QUIC stacks such as QUIC-GO.

In [13], the authors analyze the performance of the CUBIC congestion control algorithm with the hystart option enabled. All results were obtained with the ns-3 simulator and a test-bed called w-ilab.t [14]. For a mobile network, the paper highlights a real performance issue in some cases. But this study does not address the hystart of the QUIC protocol nor does it measure the impact of this option on fairness.

C. Characterization of mobile networks in transport protocol analysis

A user's data rate on a mobile network can be affected by several phenomena, one of them being the quality of the signal received by the user's equipment (UE). Signal quality depends on multiple parameters such as the UE - antenna distance, the presence of obstacles between them and interference. Typically, a large UE - antenna distance may significantly reduce the data rate. Another well-known phenomenon is the cell load. Indeed, the data rate allocated to the UE depends directly on this load [15] as a cell shares its resources between the active users.

The performances of TCP in mobile networks is a topic widely covered in the literature, for instance in [16], [17]. In [18], the performances of congestion controls are evaluated with TCP and also QUIC. These evaluations rely on the mobile network emulator Mahimahi [19], [20] that we also used (see Section III). However to our knowledge, the fairness of QUIC in mobile network has not been specifically analyzed in the literature.

Similarly, mobile network delays are caused by several phenomena and can be split between delays on the air interface, in the base-station, in the operator wire-line network and server-location dependent Internet delays. According to [21] the typical values for these delays are respectively in the range of $\approx$ 25ms, $\approx$ 30ms and $\approx$ 50ms for a server in the same country. This adds to the queuing and scheduling delays in the base station which typically depend on the per-user buffer size and on data rate variation. In current engineering practices, buffer sizing (BS) is done according to: $BS = D * \text{RTT}_{\text{EE}}$ where D is the highest supported data rate and RTT_{EE} is the typical end-to-end round trip time. Variations in the radio link data rate produce variations in queuing delays. As a numerical example, consider a buffer sized for a typical 50ms end-to-end RTT and a highest supported data rate of 100Mbps. An elastic traffic source (e.g. governed by CUBIC) that tends to fill such a buffer will see its queuing delay raise to 500ms when the radio link speed decreases down to 10Mbps.

The so-called "buffer-bloat" phenomena has been observed on cellular networks in several studies (e.g. [22], [23]) which are consistently reporting RTT values between 500ms and 2s with bulk data transfer from a TCP server loading the downlink. The buffer sizes used for the tests further described in this paper (see Section V) are consistent with these observations.

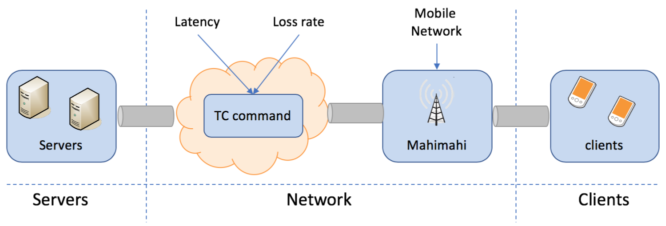


Fig. 2: Test-bed architecture

III. Fairness Assessment Test-bed

To evaluate the fairness achieved between QUIC and TCP when they compete in a network bottleneck, we implement a test-bed platform which emulates various network conditions corresponding to real mobile traffic scenarios. As shown in Fig. 2 the test-bed considers a client-server architecture and a network emulator. We employ real (non-simulated) TCP- and QUIC-based content delivery servers with associated clients. An Long Term Evolution (LTE) network is emulated with the combination of the tool Mahimahi [19], [20] (see III-B) which reproduces the transmission opportunities observed on a real LTE network and the tool *tc* which generates a given latency and packet loss level. Fairness measurements are performed during file transfers.

A. Test-bed settings

We respectively use *netcat* on Linux for TCP and *QUIC-GO* [3], [8] open-source distributions for QUIC clients and servers implementations. These applications are each executed on Linux-based kernel packaged in Ubuntu 16.04. We specifically use the same congestion control (namely CUBIC) for both TCP- and QUIC-based connections. TCP and its congestion control algorithm are both implemented in the OS kernel.

Several default TCP parameters are modified because they are not currently implemented in QUIC which is still a rather young protocol in the fields of acceleration and optimization. Firstly, the TCP Segmentation Offload (TSO) of TCP is disabled. TSO aims at reducing the CPU load, by moving the data segmentation from the kernel to the network card. This data segmentation allows data to be sent over a network in the ideal size. The QUIC transposition of TSO, named Generic Segmentation Offload (GSO) [24], is not yet supported by currently available implementations. Secondly, the Linux kernel parameter *tcp_no_metric_save* (which do not have an equivalent in available QUIC implementations) is enabled to make successive tests independent from one another.

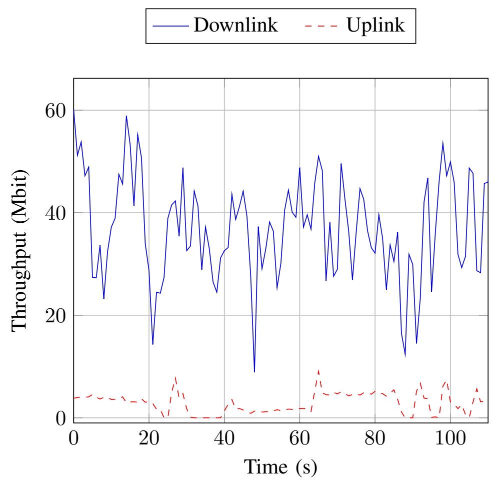


Fig. 3: Mahimahi packets delivery trace

B. Packet delivery trace files

The "link shell" of *Mahimahi* uses packet delivery trace files in order to emulate the uplink and downlink characteristics of a LTE network in terms of transmission opportunities. Each line in a packet delivery trace represents the time (at the millisecond timescale) at which a 1500 byte packet had the opportunity to be transmitted.

Packet delivery trace files are acquired by using the saturator tool [25] which saturates the network path between the saturator client and the saturator server in the uplink and downlink directions and times packet deliveries. The algorithm of the tool ensures that the queue of the bottleneck is kept sufficiently filled. Each 1500 byte packet transmission opportunity provided by the network is then used (and not wasted) as the queue is never empty. The timestamps carried by the packets allow the tool to keep track of each milliseconds where one or several transmission opportunities occurred.

For the tests reported in the present paper, we captured a packet delivery trace in static conditions (the LTE smartphone hosting the client was not moving), in rather good radio link condition with a mean Channel Quality Indicator (CQI) of 10 (out of 15). The throughput variations in the packet delivery trace are shown on Fig. 3. The average downlink and uplink throughput respectively are 34 Mbps and 4 Mbps.

IV. Session fairness assessment

Several fairness metrics have been proposed in the literature, the most well-known and used metric is the Jains index defined in Equation (1).

$$J = \frac{(\sum_{k=1}^{n} x_k)^2}{n \sum_{k=1}^{n} x_k^2} \tag{1}$$

This metric can be used for evaluating fairness among various flows (e.g, application services) when each of them requires a certain data rate. The Jains index concretely compares the obtained data-rate with the optimal required data-rate according to the max-min fairness principle [26]. The drawback of this metric is that it evaluates the fairness at a given instant t while the user QoE results from what happens during the whole session.

A generic metric to measure fairness over an entire session is defined in [6]. The Session Fairness Assessment (SFA) evaluates fairness during a session while continuously taking into account dominance between the two competing flows.

To be more specific, SFA splits the competition time of two flows A and B into S slots of 100 milliseconds. For each time slot i, SFA assesses the fairness index J_i using the Jains index J as follows $J_i = 1 - J$. Fully fair and unfair cases are respectively represented by $J_i = 0$ and $J_i = 0.5$.

It also determines the dominant flow during the *i-th* slot; a flow is dominant during the *i-th* slot when it obtains more throughput than the other during this period of time). This is , characterized by D_i: $D_i = -1$ (respectively $D_i = 1$) when A (respectively B) is dominant during the *i-th* slot.

Then, SFA defines a global and single value F to determine the *fairness* level achieved during a session while averaging the product of J_i and D_i values as expressed in Equation (2):

$$F = \frac{\sum_{i=0}^{S} D_i * J_i}{S} \tag{2}$$

V. Fairness analysis

We perform an exhaustive combination of network scenarios (referred in the following as *tests*) according to the parameters shown in Table I.

TABLE I: Network and session characteristics

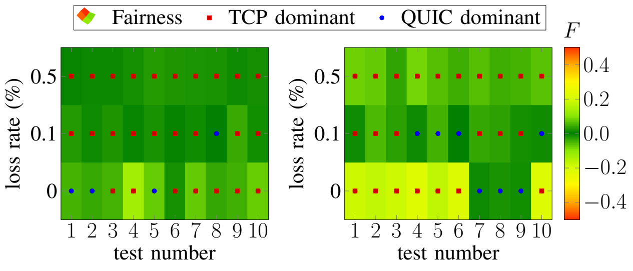
Latency (ms)	50	300	
Loss rate (%)	0	0.1	0.5
Buffer size (Mbytes)	0.750	4.5	
N **(emulated TCP connection)**	1	2	5

Latency and loss rate values are determined according to values found on Orange operational networks. As for the buffer sizes (namely BS), they are set to twice the Bandwidth-Delay Product (BDP) as stated in equation 3. The BDP corresponds to the multiplication of RTT by the maximum data rate found in the packet delivery trace (D_{max}). In our case, we assume that the RTT corresponds approximately to twice the added latency (L).

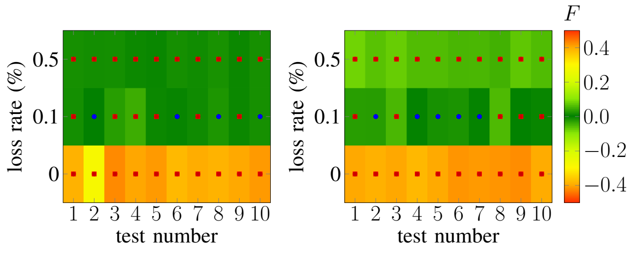
$$BS = 2 * L * D_{max} \tag{3}$$

Servers send a 100MB file which corresponds to a minimum period of 60 seconds for each test. Each test is repeated 10 times.

All results are presented in the form of color maps. The x-axis is the test number and the y-axis is the loss rate simulated in the network emulator. The colors indicate the fairness achieved between the protocols. A green color indicates that the results are fair (metric close to 0) whereas a red color indicates that the results are unfair (metric close to 0.5 or -0.5). For each test, a red square indicates that TCP is dominant (obtain more data rate during the session), whereas a blue circle indicates that QUIC is dominant.

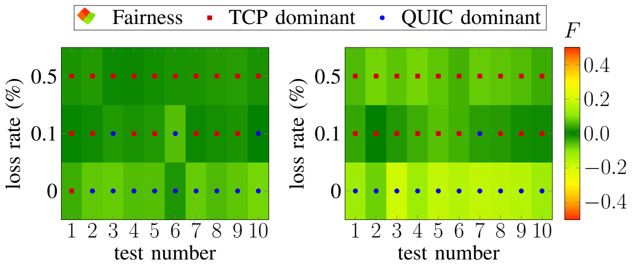


(a) Latency 50ms, Buffer size 750kB (b) Latency 300ms, Buffer size 750kB

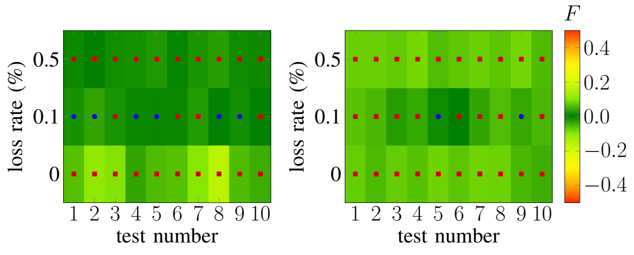


(c) Latency 50ms, Buffer size 4.5MB (d) Latency 300ms, Buffer size 4.5MB

Fig. 4: Evaluation of fairness when a single TCP flow starts 8sec before a single QUIC flow with $N = 1$ and both protocols have the hystart option enabled

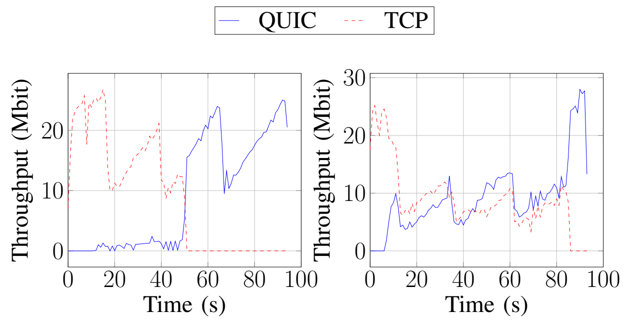


(a) Latency 50ms, Buffer size 750kB (b) Latency 300ms, Buffer size 750kB



(c) Latency 50ms, Buffer size 4.5MB (d) Latency 300ms, Buffer size 4.5MB

Fig. 5: Evaluation of fairness when a single TCP flow with the hystart option enabled starts 8 seconds before a single QUIC flow with $N = 1$ and the hystart option disabled.



(a) hystart problem on the second flow (b) data rate evolution without hystart problem

Fig. 6: Effects of QUIC hystart on data rate

A. Impact of the hystart option

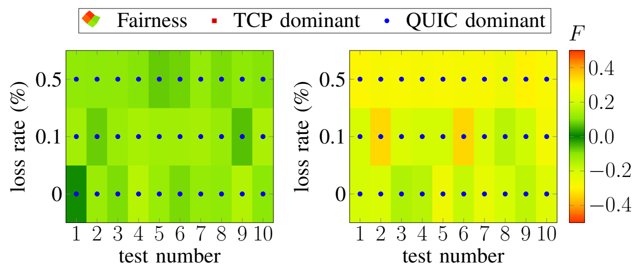
In this section, we analyze the fairness achieved between a single QUIC flow emulating a number of TCP connections equal to 1 ($N = 1$) and a single TCP flow. To perform the measurement, all QUIC and TCP settings remained by default except those that were disabled as explained in Section IV. To check that QUIC is able to obtain a fair share of the data rate when a TCP flow is started first, the TCP flow begins data transmission 8 seconds before the QUIC flow. We realize this time delay to verify that the QUIC flow succeeded in obtaining a share equivalent to TCP flow which occupies the entire bandwidth. The results represent all the tests described in the table I and we repeat all tests 10 times. Then, the fairness is assessed with the SFA method.

The fairness results are given in Fig. 4. With large buffer sizes (4.5MB) there is a fairness issue when the loss rate is set to 0% on the network emulator. This is shown on Fig. 4c and 4d. In these network situations, the average fairness values are close to 0.40, which indicates session unfairness. This unfairness is illustrated in Fig. 6a which shows how transmission opportunities are shared during a representative test. During the first 8 seconds, the TCP flow uses all the available bandwidth because it is the only one on the network. After these first 8 seconds, the QUIC flow starts. However, the intended slow-start phase of the QUIC protocol is missing, which prevents a fast growth of the QUIC congestion window. The QUIC flow is able to use a significant share of the bandwidth only when the TCP flow has finished (at $\approx$ 50s).

To check the impact of the hystart option, we disable the hystart option on the QUIC flow. The new test results are shown on Fig. 5. There is no more unfairness between QUIC and TCP. This is also visible on Fig. 6b which shows the data rate evolution for the two flows with the same settings (no hystart option for QUIC). It is seen that, at the beginning of the QUIC connection, a rapid increase in data rate is due to a slow-start phase. This larger increase in data rate yields a fair sharing of data rate between the 2 flows.

In order to understand how hystart may yield unfairness, keep in mind that the exit of the slow-start caused by the hystart option is due to an increase in the RTT. Several phenomena can explain the hystart triggering:

- *Increased buffer filling:* During the start of the QUIC flow, the TCP flow continues to increase its data rate as long as no packet loss occurs. Having very few packet losses on the network, this has the effect of increasing



(a) 1 TCP, 1 QUIC N=2 (b) 1 TCP, 1 QUIC N=5

Fig. 7: Evaluation of fairness with $N = 1, 2, 5$, buffer size $750kB$, hystart disabled when the TCP flow starts 8s before the QUIC flow.

the buffer filling and thus increasing the RTT. So, when the QUIC flow is in the slow-start phase, the TCP flow will increase the RTT of the QUIC flow.

- *Decrease in data rate:* The start of QUIC flow is not perfectly synchronized with the packet delivery trace file. Thus, the flow can start either during a network capacity increase phase or during a network capacity decrease phase. Thus, when the flow starts on a decreasing network capacity phase, it results in an increase in the RTT.

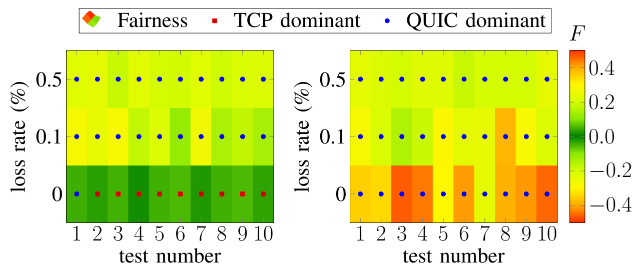
None of the above phenomena is indeed related to an incipient congestion, although this is how both are (wrongly) interpreted by hystart which throttles the QUIC flow.

B. Impact of the number of emulated simultaneous TCP connection in QUIC

In this section, we evaluate the fairness between a TCP connection and a QUIC connection with a varying number of emulated TCP connections in QUIC. The hystart option is disabled for both protocols. To present the results, we consider a 750kB buffer size and a 50ms network latency. The other configurations have the same behavior except for a buffer of 4.5MB and $N = 5$ which is analyzed in Section V-C. The test results are shown in Fig. 7.

In Fig. 5a, a TCP flow competes with a QUIC flow emulating a single TCP connection. Fairness is achieved. The average of fairness indices is close to 0. In Fig. 7a and 7b, when N increases, the fairness index decreases. Indeed, when N is equal to 2 the average fairness index is close to 0.1, and when the value of N is equal to 5 the average fairness index is close to 0.3, which characterizes a high degree of unfairness.

This is straightforwardly explained by the impact of N on the evolution of the congestion window size over time. Indeed, it has been shown in [6] that N mainly impacts on the decrease of the congestion window after a loss event: in case of packet loss when $N = 1$ (respectively $N = 5$) the congestion control algorithm decreases the congestion window size by 30% (respectively 5%). In other words, in test scenarios with values of N larger than 1, the congestion window reduction due to packet loss is less severe for QUIC than for TCP, which implies that QUIC can maintain a higher throughput than TCP even if they are submitted to the same packet loss rate.



(a) 1 TCP, 1 QUIC N=5 default CWS_{max} (1000 packets) (b) 1 TCP, 1 QUIC N=5 increased CWS_{max} (10000 packets)

Fig. 8: Evaluation of fairness with $N = 5$, buffer size $4.5MB$, hystart disabled when the TCP flow starts 8s before the QUIC flow: issue with the default CWS_{max}.

C. Impact of maximum congestion window size

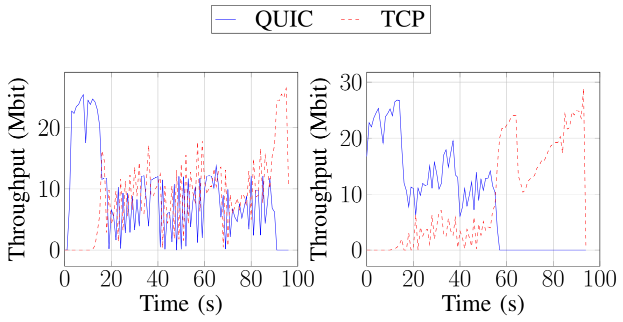
In this section, we highlight the impact on fairness of another parameter characterizing CUBIC. We focus on the following network configuration: a buffer size equal to 4.5MB, latency equal to 300ms and $N = 5$ for QUIC. Results with a smaller network latency (50ms) are quite similar to those discussed below.

Fairness results are shown in Fig. 8; the two sets of tests differ by different values for the upper bound CWS_{max} set for the congestion window size in QUIC. The default value for CWS_{max} is respectively set to 1000 and 2000 packets in the QUIC-GO and Chromium implementations. It is important to note that this value does not change over session nor does it depend on N. Also, upper bounding the congestion window size is not related to flow control. Flow control limits the source rate so as not to overload the receiver. Being on high-capacity computers, the current study is not impacted by rate limitations due to flow control.

In Fig. 8a, we see a significant unfairness except for a 0 packet loss. As explained in Section V-B, for $N = 5$ and a some packet loss, unfairness is indeed to be expected as the rate reduction for QUIC is more limited than for TCP in case of packet loss. When the packet loss ratio is set to zero on the LTE network emulation, the only packet losses correspond to those occurring in the bottleneck buffer.

We observe that unfair sharing, even for 0 packet loss, is visible in Fig. 7b, whereas it is not the case in Fig. 8a; test conditions only differ in terms of bottleneck buffer size. In order to understand what happens when the bottleneck buffer size is large, a representative data rate evolution in this case is shown in Fig. 9a.

Fig. 9a shows that both TCP and QUIC obtain roughly the same throughput. This is because the default (small) CWS_{max} value, the maximum congestion window size for QUIC, is quickly reached in a 0 packet loss network. As the buffer size in Fig. 8a is quite large (4.5MB), whereas the throughput for QUIC is frozen to the value corresponding to CWS_{max}, the TCP flow can still increase its own congestion window and therefore its throughput, as long as the bottleneck buffer does



(a) 1 QUIC $N = 5$ default CWS_{max} (1000 packets), 1 TCP

(b) 1 QUIC $N = 5$ increased CWS_{max} (10000 packets), 1 TCP

Fig. 9: Effects of limitation of congestion window size, when the loss rate equal to 0%

not overflow. Buffer overflow, yielding unfairness, occurs more quickly for a smaller buffer as in Fig. 7b.

In order to confirm that it is the combination between a large buffer size and a small CWS_{max} that leads to the observed situation, we consider the same parameters as those used for Fig. 8a, except for CWS_{max} which is now set to a ten times higher value ($CWS_{max} = 10000$). Fairness results are shown in Fig. 8b. In this new QUIC configuration, unfair sharing between TCP and QUIC is clearly visible even if no packet loss is simulated.

Fig. 9b shows a representative example of the data rate evolution when the loss rate is equal to 0% and the maximum size of the QUIC congestion window is equal to 10000. It shows that the TCP flow is starved by the QUIC flow which can now increase significantly its congestion window.

VI. QUIC: The law of the strongest in a jungle of capacity sharing behaviours ?

As we have seen in Section V, the developers of QUIC open source implementations [8], [9] have their own views on what is the fair sharing of network capacities with TCP flows. On the deployment front, it is cumbersome to reverse engineer the fairness policy followed by QUIC endpoints and to the best of our knowledge such policies are undocumented by implementers, including Google with its server-side QUIC stack (the code and specific tunings of Google's QUIC server are currently kept secret [5]). Moreover, today these issues are largely ignored in QUIC standardization [3].

With its anti-ossification features, QUIC is designed to be easier to evolve and re-deploy than it's predecessor TCP. This favours the fast emergence of QUIC variants. Additionally, deploying a QUIC variant is accessible to application editors (as opposed to operating system editors for TCP). This is a favourable ground to see significantly more QUIC variants being actually deployed than in TCP history.

Is this opening a new Internet era, in which the policies governing the fair sharing of network capacities are no longer resulting from a collective agreement and in which a multitude of varying individual choices will be made ?

In the authors' opinion, if it is conceivable that some fairness (e.g. as measured by the SFA defined in Section IV) can be traded for a better utilization of network capacities or for an improved user-experience (these two aspects being out of the scope of the current paper), such trade-offs should however be the result of open discussions and collective agreements in the Internet community, including the relevant regulation bodies, rather than the result of the law of the strongest.

We advocate here that the monitoring and documenting of new practices in network capacity sharing between concurrent flows should fuel such discussions, and generate implementation guidelines by regulation bodies.

VII. Conclusion

QUIC is currently being deployed on the Internet and is intended to replace TCP and TLS. It adopts several options already implemented in TCP such as hystart and CUBIC. However, its implementation must not be done at the expense of user's fairness, this is a growing concern for operators as their customer services are first in line when an internet application performance drops.

In the work reported in the present paper, we have analyzed how QUIC implementations can impact fairness when competing with a TCP flow on the wireless link of a mobile networks. To achieve this fairness measurement, both protocols use the same congestion control algorithm (CUBIC) in order to isolate the impact of protocol implementations on fairness. We used a laboratory test-bed to measure the behaviour of each of the TCP and QUIC flows. Several aspects are analyzed during the measurements and we highlight several fairness issues. Firstly, the impact of hystart was measured on the different flows in different situations on the mobile network. Then we performed a fairness measurement with a varying number (N) of TCP connections emulated in QUIC. Finally, we studied the impact on fairness of the limitation of the congestion window size in QUIC for large values of N.

As regards the hystart option present in TCP and QUIC implementations, the results clearly show a significant network capacity sharing issue. This problem occurs mainly when loss rates on the network are low. Indeed, in some network configurations, this option prevents the slow start from operating as it should, i.e. facilitating a fast ramp-up of the throughput either at launch time, or after a timeout of the connection. The behaviour of the hystart option may therefore considerably reduce the user's data rate during the entire session and thus create a real problem in terms of QoE. We thus recommend that this option be disabled or significantly improved (by not systematically interpreting RTT increase as incipient congestion) on QUIC and TCP, to ensure good network utilization and fairness.

These results reported here show that the value of N has a real impact on the behaviour of QUIC's congestion control algorithm and on the fairness between protocols. In our previous study [6], we demonstrated that this problem is already present on fixed networks. Here we have shown that the same problems are visible on mobile networks. When $N = 1$, the TCP and QUIC protocols fairly share available bandwidth in all the network configurations tested. But when the value of N increases then the fairness decreases considerably in favor of QUIC. That is to say, the QUIC protocol obtains a much higher share of throughput than the TCP protocol. The setting of N is currently static in QUIC implementations and is by default set to 2. However, it can be easily modified and incremented to larger values in client and server applications. It is therefore essential to be able to define engineering rules to dynamically adapt this value over time according to user requests (e. g. number of streams in QUIC) and network conditions to ensure a fair behavior of protocols on the network. We advocate that implementation guidelines and monitoring of usages would be beneficial to the majority.

The paper also studied the impact of the limit set to the congestion window size by the CUBIC congestion control algorithm. For QUIC, this value is constant during the session and does not depend on N. We show that when the values of N increase, the limitation of the congestion window size prevents the QUIC protocol from increasing its throughput. This limitation therefore does not allow the emulation of several TCP connections when the loss rate on the network is low. This limitation hinders the principle of multiplexing multiple application connections in the QUIC protocol.

Lastly, we pointed out that our results were obtained considering available, open source versions of QUIC whereas it is highly likely that most of the current QUIC traffic is transported over closed source versions operated by Google. This advocates in favour of more effort being deployed by standards and regulation bodies to provide specifications and guidelines for future transport protocols, including QUIC.

VIII. Acknowledgement

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

References

[1] A. Gupta and R. K. Jha, "A Survey of 5G Network: Architecture and Emerging Technologies," *IEEE Access*, vol. 3, pp. 1206–1232, 2015.

[2] A. Langley and et al., "The QUIC Transport Protocol: Design and Internet-Scale Deployment," in *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*, ser. SIGCOMM '17. New York, NY, USA: ACM, 2017, pp. 183–196.

[3] J. Iyengar and M. Thomson, "QUIC: A UDP-Based Multiplexed and Secure Transport," IETF, Internet-Draft draft-ietf-quic-transport-20, 2019.

[8] QUIC-GO. [Online]. Available: https://github.com/lucas-clemente/quic-go

[4] R. R. Stewart, "Stream Control Transmission Protocol," RFC 4960, Sep. 2007. [Online]. Available: https://rfc-editor.org/rfc/rfc4960.txt

[5] 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 *Proceedings of the 2017 Internet Measurement Conference*. ACM, 2017, pp. 290–303.

[6] R. Corbel, S. Tuffin, A. Gravey, X. Marjou, and A. Braud, "Assessing the impact of QUIC on network fairness," in *2nd International Conference on Communication and Network Protocol (ICCNP 2019)*, 2019.

[7] S. Ha and I. Rhee, "Hybrid slow start for high-bandwidth and long-distance networks," in *Proc. PFLDnet*, 2008, pp. 1–6.

[9] Chromium. [Online]. Available: https://github.com/chromium/chromium

[10] proto-QUIC. [Online]. Available: https://github.com/google/proto-quic

[11] S. Ha and I. Rhee, "Taming the elephants: New TCP slow start," *Computer Networks*, vol. 55, no. 9, pp. 2092–2110, 2011.

[12] I. Rhee, L. Xu, S. Ha, A. Zimmermann, L. Eggert, and R. Scheffenegger, "CUBIC for Fast Long-Distance Networks," RFC 8312, Feb. 2018. [Online]. Available: https://rfc-editor.org/rfc/rfc8312.txt

[13] E. Atxutegi, Å. Arvidsson, F. Liberal, K.-J. Grinnemo, and A. Brunstrom, *TCP Performance over Current Cellular Access: A Comprehensive Analysis*. Springer International Publishing, 2018, pp. 371–400.

[14] S. Bouckaert, W. Vandenberghe, B. Jooris, I. Moerman, and P. Demeester, "The w-iLab. t testbed," in *International Conference on Testbeds and Research Infrastructures*. Springer, 2010, pp. 145–154.

[15] B. Nguyen, A. Banerjee, V. Gopalakrishnan, S. Kasera, S. Lee, A. Shaikh, and J. Van der Merwe, "Towards understanding tcp performance on lte/epc mobile networks," in *Proceedings of the 4th workshop on All things cellular: operations, applications, & challenges*. ACM, 2014, pp. 41–46.

[16] F. Ren and C. Lin, "Modeling and improving tcp performance over cellular link with variable bandwidth," *IEEE Transactions on Mobile Computing*, vol. 10, no. 8, pp. 1057–1070, Aug 2011.

[17] F. Li, J. Won Chung, X. Jiang, and M. Claypool, *TCP CUBIC versus BBR on the Highway*, 03 2018, pp. 269–280.

[18] F. Y. Yan, J. Ma, G. D. Hill, D. Raghavan, R. S. Wahby, P. Levis, and K. Winstein, "Pantheon: the training ground for internet congestion-control research," in *2018 USENIX Annual Technical Conference (USENIX ATC 18)*. Boston, MA: USENIX Association, 2018, pp. 731–743. [Online]. Available: https://www.usenix.org/conference/atc18/presentation/yan-francis

[19] R. Netravali, A. Sivaraman, S. Das, A. Goyal, K. Winstein, J. Mickens, and H. Balakrishnan, "Mahimahi: Accurate Record-and-Replay for {HTTP}," in *2015 {USENIX} Annual Technical Conference ({USENIX}{ATC} 15)*, 2015, pp. 417–429.

[20] Mahimahi. [Online]. Available: https://github.com/ravinet/mahimahi

[21] P. Deb, Modiano and et al., "NSF Workshop on Ultra-Low Latency Wireless Networks," Mar. 2015.

[22] H. Jiang, Z. Liu, Y. Wang, K. Lee, and I. Rhee, "Understanding bufferbloat in cellular networks," in *Proceedings of the 2012 ACM SIGCOMM workshop on Cellular networks: operations, challenges, and future design*. ACM, 2012, pp. 1–6.

[23] S. Alfredsson, G. Del Giudice, J. Garcia, A. Brunstrom, L. De Cicco, and S. Mascolo, "Impact of TCP congestion control on bufferbloat in cellular networks," in *2013 IEEE 14th International Symposium on" A World of Wireless, Mobile and Multimedia Networks"(WoWMoM)*. IEEE, 2013, pp. 1–7.

[24] W. de Bruijn and E. Dumazet, "Optimizing UDP for content delivery: GSO, pacing and zerocopy."

[25] Saturator. [Online]. Available: https://github.com/keithw/multisend/blob/master/sender/saturatr.cc

[26] J.-Y. Boudec, "Rate adaptation, Congestion Control and Fairness: A Tutorial," 2000.