QUIC: Opportunities and threats in SATCOM

Nicolas Kuhn *CNES*
François Michel *UCLouvain*
Ludovic Thomas *EPFL*
Emmanuel Dubois *CNES*
Emmanuel Lochin *ENAC*

***Abstract*—This article proposes a discussion on the strengths, weaknesses, opportunities and threats related to the deployment of QUIC end-to-end from a satellite-operator point-of-view. The deployment of QUIC is an opportunity for improving the quality of experience when exploiting satellite broadband accesses. Indeed, the fast establishment of secured connections reduces the short files transmission time. Moreover, removing transport layer performance enhancing proxies reduces the cost of network infrastructures and improves the integration of satellite systems. However, the congestion and flow controls at end points are not always suitable for satellite communications due to the intrinsic high bandwidth-delay product.**

Further acceptance of QUIC in satellite systems would be guaranteed if its performance in specific use-cases is increased. We propose a running code for an IETF document, and based on an emulated platform and on open-source software, this paper proposes values of performance metrics just as one piece of the puzzle. The final performance objective requires consensus among the different actors. The objective should be challenging enough for satellite operators to allow QUIC traffic but reasonable enough to keep QUIC deployable on the Internet.

***Index Terms*—QUIC, SATCOM, PEP**

I. Introduction

While its standardization at the Internet Engineering Task Force (IETF) is still an on-going activity [1], QUIC is already deployed by several companies. Back in 2018, Google QUIC was already exploited in Google and Facebook services [2], [3]. Several web hosting companies such as Akamai, CloudFare or Fastly are involved in the standardization process and develop a QUIC server stack. On the client side, Firefox and Google Chrome are ready for initiating QUIC connections.

The QUIC IETF working group aims for inter-operable implementations but is not expected to define congestion control solutions. The adequate congestion control depends on application needs and most mechanisms do not need to be standardized to guarantee inter-operability. The current QUIC specification proposes TCP NewReno as congestion control [4]. Even if not specified, newer mechanisms such as BBR [5] are undoubtedly already deployed in several QUIC implementations (picoquic [6], quiche [7], Chromium [8]). In addition to this variety of congestion control algorithms, updates of QUIC in end-point stacks are frequent [2]. As a consequence, it is hard to publish relevant protocol performance [2].

QUIC's control information is part of its encrypted and authenticated data, making it impossible for a third party to intercept a QUIC connection similarly to what Performance Enhancing Proxies (PEP) [9] are currently doing with TCP. As a consequence QUIC traffic will be seen as standard UDP traffic, will not benefit from PEP optimization and could be either classified as non-congestion controlled traffic by the PEP QoS policy, shaped or dropped. The important research activity related to the evaluation and the adaptation of TCP and HTTP over SATCOM systems [10]–[12] needs to be revisited with the impact of the deployment of applications using QUIC.

When comparing systems exploiting TCP proxies or QUIC, the secured connection establishment of QUIC saves one round-trip as opposed to a classic TCP+TLS connection establishment. QUIC can then achieve fairly good performance for short files. However, QUIC does not outperform TCP proxy solutions for large pages [13], [14] in satellite broadband systems. Since QUIC is encapsulated in UDP, it does not take benefit from TCP PEP optimizations. On a side note, the performance evaluation of web services is complex and depends on the web page characteristics [15]; which makes it hard to guarantee that performance would be improved, or not, by using QUIC as opposed to what is currently deployed.

There is a risk that operators block QUIC because it questions the way operators manage their networks and guarantee the SLA and transport-layer performance to their customers.

- This paper proposes a discussion on the opportunities and threats brought by the deployment of QUIC on satellite broadband systems.

Further, we can assume that QUIC acceptance in satellite systems would be guaranteed if it achieves good performance in specific use-cases. While there is a current IETF document identifying how to ensure acceptable protocol performance [16], there is no actual evaluation framework available.

- This paper also proposes running code and an evaluation framework, so that anyone can contribute to the performance objectives of QUIC.

The paper is organized as follows: in Section II-A, based on Google QUIC performance evaluations on a public satellite access, we compare the performance of a context-agnostic congestion control and an optimized TCP proxy. In Section II-B, we also identify the E2E losses that are experienced in a public SATCOM access. We then exploit a satellite system in a controlled environment to assess the impact of these losses on TCP connections in Section II-C. In Section III, based on the experimental results of Section II, we propose a Strengths, Weaknesses, Opportunities, Threats (SWOT) analysis for QUIC over satellite from a satellite-operator point of view. In Section IV, using an emulated platform and open-source material, we propose a framework for defining

performance objectives for QUIC over different satellite-based broadband systems. We also propose resulting example values that could be discussed among the different actors. Section V describes some hints on how QUIC can be adapted to fulfill the objectives that the community could propose.

II. Challenges for QUIC in SATCOM

This section identifies some issues that QUIC encounters over satellite networks. We exploit a public satellite access to (1) compare the performances of GQUIC with those of its TCP-proxy counterpart (in Section II-A) and to (2) measure the E2E losses experienced without local recovery (in Section II-B). To identify the impact of losses on an E2E connection, we assess the impact of random losses on a controlled satellite network (in Section II-C).

A. GQUIC performance on satellite public access

The rationale of this subsection is to compare the performance of Google QUIC with those of its TCP-proxy counterpart to further identify issues that QUIC faces in satellite systems. The results presented in this section exploit a public SATCOM access operating in Europe with geostationary satellites. We use a Eutelsat KA-SAT PRO25Go access.

We focus on the browsing experience using the Chrome browser with simple pages accessible with both TCP and GQUIC. Chrome browser supports GQUIC [17]. We consider two targets A and B with a respective size of 5.3 MB and 11 kB. These pages are composed of less than 2 objects. We do not aim a wide comparison between GQUIC and TCP for web services since this depends on the characteristics of the web pages [15] but rather aim at identifying trends in the results. We focus on the page load time (PLT) [18] metric as it returns similar trending results to visual QoE metrics for such simple pages.

The testbed is shown in Figure 1. Using RTT measurements conjointly with the `traceroute` tool, we have identified that the public SATCOM access uses a distributed transparent PEP architecture for TCP connections: proxies are located both at the gateway and at the satellite terminal. We cannot obtain much more information on the proxies stacks used within the operator network nor within the satellite terminal. However, the PEPs cannot split the GQUIC connections as most of their control information is part of the QUIC encrypted data.

Similarly, the setup of the experiments is as fair as possible for both protocols. In particular, Google has deployed the BBR congestion control [5]: we request therefore the same congestion control for GQUIC, using flags during the handshake [19] although we do not control the network stack of the remote server. Tests are performed using the Selenium automation tools [20] and they have been designed based on Chrome behavior `67.0.3396.99` (see [13], [21] for more details).

Figure 2 presents the distribution of the PLTs for each protocol and each target. We observe that GQUIC performs slightly better than TCP for target B, with a smaller dispersion. However, GQUIC performs significantly worse than TCP for

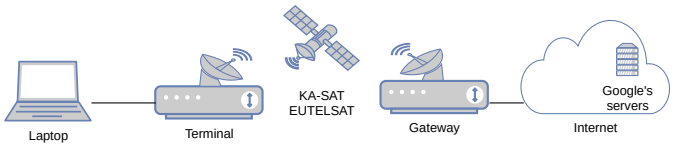

Fig. 1. Testbed for GQUIC performance assessment

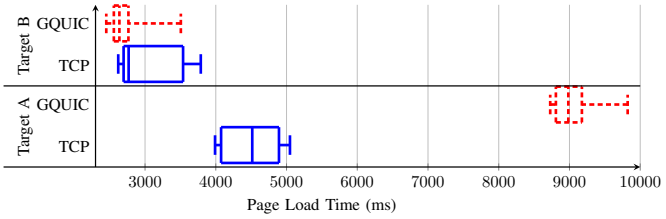

Fig. 2. Percentiles of the PLT over 40 tests

the heavy target A. The HTTP GET messages using GQUIC have a delay approximately twice as long as with TCP.

The small gain of GQUIC over TCP for the small target B can be explained by GQUIC requiring less RTTs than the pair TCP/TLS to establish both transport- and secure-layer connections. Yet the gain is less than one RTT. For example, due to the distributed PEP architecture, the TCP handshake of the laptop is performed locally with the satellite terminal and does not require a full E2E RTT: from the client viewpoint, the TCP connection is established before it actually is with the server. Moreover, the communication between the PEP at the terminal and the PEP at the satellite gateway may use already-established connections. The TLS connection establishment can start faster than in classical E2E TCP+TLS connections.

To explain the qualitative difference for target A, we present in Figure 3 the evolution of the sequence numbers for both TCP and GQUIC. Their evolution is displayed over the course of the connection since the client sent the TCP SYN

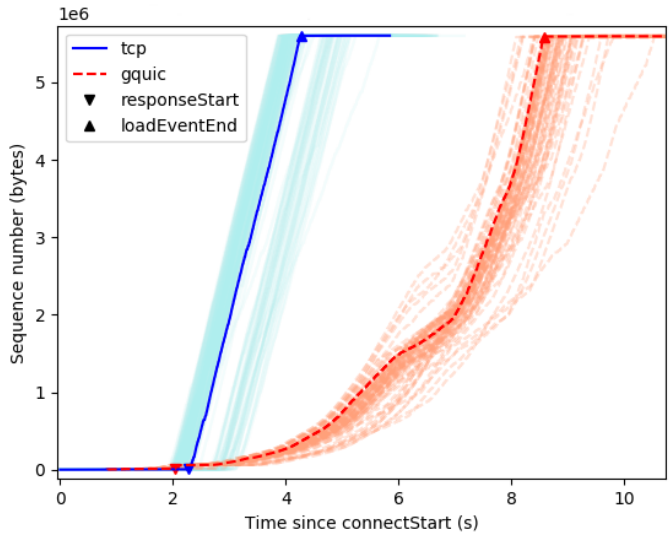

Fig. 3. Sequence number evolution for target A. Focus on 2 out of 40 tests.

or GQUIC ClientHello packet. We first note that the event `responseStart` (when the first byte of the HTTP answer is received) is fired earlier using GQUIC than using TCP, which emphasizes again that the connection establishment is faster with GQUIC than with TCP. However, GQUIC then increases its throughput slowly, and the BBR slow start requires up to 4 seconds to probe the whole E2E link. On the contrary, TCP achieves a constant throughput in less than 50ms. This value is consistent with the fact that the server TCP stack only needs to probe the link to the satellite operator network because the PEP in the satellite gateway terminates the connection.

Even though the results are only valid for GQUIC, they point out the challenges faced by QUIC in a satellite context. On one hand, reducing the handshake duration has a positive effect on the performance, especially for small objects. On the other hand, enforcing to probe the entire link has a negative effect on the performance, especially for medium-size objects.

B. Losses in public SATCOM access

In this section, we measure the end-to-end loss ratio that will be experienced without local recovery by using a public satellite broadband access. The testbed used is the same as in Figure 1 except we now manage the server which is located at ISAE-SUPAERO[1]. The client laptop is still located at CNES[2]. The QUIC implementation is `picoquic` [6], more specifically its PQUIC [22] fork. We study the losses experienced using satellite links under several conditions. Two experiments have been performed with different technology to bridge the laptop and the satellite terminal: a wired access and a WiFi access point. Two hundred experiments have been run with the wired access to the satellite terminal. During each experiment, one 1 MB file is downloaded. Packets are captured on both client and server, which we both control. We count the number of packets sent by the server that never arrived to the client. Three hundred other experiments have been performed with the WiFi access point. Because of time restrictions on the shared satellite access, only 500 kB files have been downloaded for these Wi-Fi experiments.

Table I shows the computed metrics. For each experiment, we report the uniform loss rate ($loss_{uni}$), the simplified Gilbert model estimated transition probabilities and the maximum burst size (B_{max}). The simplified Gilbert model transition probabilities are expressed in the notation $P(s_1|s_2)$ proposed in [23] where s_1 (resp. s_2) is the destination (resp. origin) state. The states are either the Good (g) or Bad (b) state. As shown on the table, the WiFi link seems to suffer from longer loss bursts and a higher uniform loss rate. While the validity of these models is questionable (*e.g.*, Markov modelisation does not apply), they provide insights on the amount of losses and the approximate size of bursts of losses that are experienced in public and shared SATCOM accesses. This helps the community in providing values that can be considered (among others) when evaluating end-to-end proposals such as QUIC.

[1]National Higher French Institute of Aeronautics and Space based in Toulouse, France

[2]French Space Agency

TABLE I
END-TO-END LOSS-RELATED METRICS FROM EXPERIMENTS WITH A WIRED AND WIRELESS ACCESS POINTS.

Method	$loss_{uni}$	$P(g\|g)$	$P(g\|b)$	$P(b\|b)$	$P(b\|g)$	B_{max}
Wired	0.017	0.983	0.935	0.065	0.017	15
Wi-Fi	0.028	0.982	0.645	0.355	0.018	47

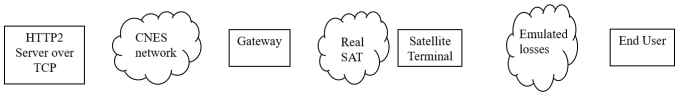

Fig. 4. Testbed for impact of losses on E2E connection assessment

C. Impact of losses on an E2E connection

This section assesses the impact of end-to-end losses of a TCP connection over a real satellite platform. The rationale is to obtain base performances that QUIC should obtained in the same conditions. We call these performances objectives : QUIC targets. The architecture used is shown in Figure 4. Both client and server run a 4.4.0-135 Linux Kernel with default parameters. The congestion control is CUBIC [24]. Random losses are enabled with `tc netem` command. The gateway and satellite terminals are tuned to provide a stable throughput. The available capacity is 10 Mbps in both directions.

The results are shown in Table II. Because the access to the satellite is shared, running a large amount of experiments was not possible. We thus download a very large file to reach a relevant average goodput. When a proportion of one over thousand packets is lost on average, the goodput is more than halved. Considering that the end-to-end losses effectively experienced may be even higher, as shown in Section II-B, these simple tests illustrate the importance of non-congestion losses on the performance of end-to-end protocols. Based on these results, we can expect that large files transfers using QUIC will be severely impacted by losses that can occur in the network.

III. SWOT MATRIX FOR QUIC

This section presents a SWOT analysis (Strengths Weaknesses Opportunities and Threats) from a satellite-operator perspective considering a potential deployment of QUIC end-to-end.

A. Strengths

We first discuss the strengths with only considering the protocol performance.

The strengths of QUIC reside in the performance improvement for small pages with the reduced amount of handshakes that are necessary to establish a secured connec-

TABLE II
IMPACT OF LOSSES ON E2E GOODPUT

Loss ratio	0	0.0001	0.0005	0.001	0.005
Time needed to download 1 GB (s)	797	935	1528	1863	7140
Goodput (Mbps)	10	8.5	5.2	4.2	1.1

tion, as shown in [13] and Section II-A. Compared to the TCP+TLS1.3 combo, QUIC saves one RTT by doing both transport and cryptography handshakes at once. The TCP Fast Open (TFO) [25] extension allows TCP to reach similar performance but still requires a TLS connection establishment. Moreover, TFO may not be enabled by default in the deployed TCP proxy. Adding the support for TFO in satellite terminals can be easily done with a software update but may have other impacts on the constrained resources of the satellite terminals. In general, pushing transport layer innovation is usually not done inside production systems, and this remains true for satellite terminals. However, QUIC 0-RTT connection establishment does not need the agreement of the proxy which is another advantage.

B. Weaknesses

Two main weaknesses can be identified when using QUIC for satellite communications. (1) QUIC does not enable local recovery of losses: a loss occurring between the end-user and the satellite gateway will be detected after one round-trip on the full E2E link. (2) The congestion control convergence is slow due to a large E2E delay. As PEPs cannot be used to transparently split a QUIC connection, the congestion window will take more time to reach the capacity of the link.

The slow congestion control convergence has a significant impact on the download time of large files as shown in Section II-C. Indeed, non congestion-induced loss events for loss-based congestion controls drastically reduce the goodput. In the case of short files, there is a higher probability for the lost packets to be located in the last packets flight. This induces an additional latency increasing the total download time.

C. Opportunities

This part discusses the opportunities provided by the deployment of QUIC and the removal of TCP proxies. We focus on aspects related to evolutivity and extensibility.

The opportunity regarding the deployment of QUIC is that satellite broadband systems may not require specific TCP proxies both on aggregation and customer segments. This would ease the deployment of new end-to-end innovations, such as TCP Fast Open [25] as only the endpoints are required to evolve. This would also reduce the price of satellite-operator networks; and increase the resiliency of mobile scenarios and improve the end-to-end security.

Another opportunity is the facility to extend QUIC. TCP has only a 40 bytes header length to store extension information. QUIC is a frame-based protocol and new frames can be defined to implement new protocol behaviors. Those frames can be of arbitrary length and can span a whole packet if needed. This allows to easily define extensions such as multipath [26] and Forward Erasure Correction [27], [28] to enhance QUIC performance for satellite communications. This approach would be very relevant for advising extensions to web hosting services deployed within the satellite-operator network or if these services specify their protocols depending on the characteristics of the network underneath.

D. Threats

We finally discuss the threats occurring when TCP proxies are removed and QUIC is not blocked in SATCOM systems.

Despite all these advantages and the opportunities, a non-negligible threat remains: the loss of control of the performance of the system as seen by the end users. By deploying TCP proxy, satellite broadband network providers can control the quality of the service brought to their customers. This ensures that contractual Service Level Agreements (SLA) can be obtained. Removing the TCP proxies could make it hard for operators to control the evolution of the protocols that may not be adapted to satellite systems. Moreover, the lack of proper management interfaces on various equipment that compose an operator network makes it hard to identify the faulty part of the network [29].

E. Summary

While QUIC exhibits interesting performances and cost reduction opportunities, its performance for long flow transfers is an issue for a wider adoption in satellite-based systems. To prevent satellite network operators from blocking this traffic, QUIC should consider satellite use-cases in the parametrization of its congestion and flow controls. The SWOT analysis proposed is built from a satellite operator point-of-view and may be extendable to other types of networks such as mobile networks that also exploit proxies.

IV. A framework for defining objectives for QUIC

Following the issues mentioned in Section III-E, guaranteeing that QUIC performs as well as a TCP-proxy based solution would limit the risk of satellite network operators blocking QUIC traffic. To this end, an IETF document describes satellite accesses scenarios against which QUIC could be tested [16]. These scenarios are simple in order to be integrated in regression tests of QUIC implementations. Regression tests need to precise values for the expected outcome of a test.

This section proposes a framework that can be exploited by others to contribute to the definition of the outcome of the regression tests. The objective is to provide running code for the scenarios proposed in [16] since the document only presents the configuration of the test and expected results.

We exploit open-source tools and expect to release the orchestration code in OpenBACH, an open source test orchestrator [30]. In the meantime, OpenBACH can be exploited to reproduce these experiments. The results shown here do not pretend to be generalized: the actual results depend on the kernel versions, the options in the proxy or the tool used to generate the data. The actual objective of performance require consensus among the different actors: the values proposed in this section are one piece of the puzzle. The objective should be challenging enough for satellite operators to allow QUIC traffic but reasonable enough to be deployable in the Internet.

A. Testbed description

This section describes the testbed that can be set up to assess the performance of end-to-end transfer over a satellite systems exploiting two transport proxies. This is shown in Figure 5.

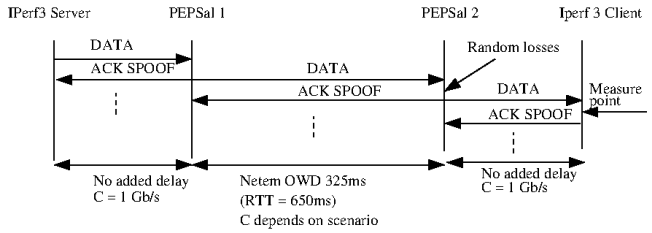

Fig. 5. Testbed for QUIC objectives assessment

TABLE III
EXAMPLES OF TCP PEP CONFIGURATIONS PARAMETERS

	TCP_WMEM_MAX (MB)	TCP_RMEM_MAX (MB)	CORE_RMEM_MAX CORE_WMEM_MAX (MB)	ICWND CWND (packets)
No PEP (purple)	4	6	0.2	10
PEP A (green)	4	6	0.2	10
PEP B (blue)	4	6	0.2	100
PEP C (orange)	33	33	33	10
PEP D (yellow)	33	33	33	100

The architecture depicts a distributed PEP configuration [9] where splitting is handle on both sides and ACK spoofing is enabled. PEPSal [31] operates to a flow control as standard commercial PEP. We will not dive into the full internal PEP mechanism implemented as this is both out of scope of this study and non mandatory for our purpose. The configuration used is the one proposed by default which roughly represents what we should expect with a commercial PEP. The link characteristics are emulated (1) by `netem` on the satellite link between PEPSal 1 and PEPSal 2 and (2) by a uniformly distributed random loss pattern applied in both directions using `tc` commands on the PEPSal 2 interface towards the iperf3 client. At the measurement point, we report the accumulated received data and the received data rate as measured by iperf3.

In [16], the authors proposed different scenarios and presented in details the amount of data generated, the required buffer sizes in bandwidth limiters and the output results for each scenario. Due to space limitation and since the goal is to offer a framework for others to contribute to the definition of objectives, the results for only two challenging scenarios are shown as examples.

End servers use a default Linux kernel version 4.15 and `iperf` 3.6 is used to generate traffic. Several Linux kernel configurations are proposed as examples and are shown in Table III. We propose to extend the memory allocated to the connection and initial congestion window to help the congestion control to get up to speed. This modifications are only applied to the PEP and not on end points to better reflect the current deployment situation. Further experiments include the application of these parameters end-to-end. The proposed framework can be used to define and experiment on other configurations. The goal is not to obtain the best performance possible but to contribute to the definition of reasonable objectives.

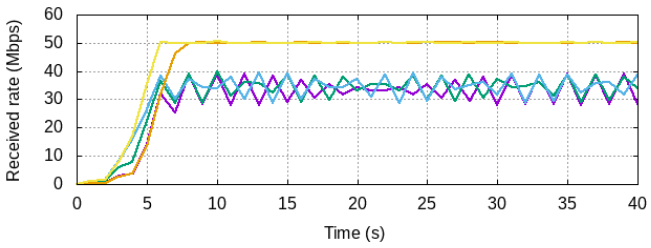

Fig. 6. Received rate for 50 Mbps / 10 Mbps use case

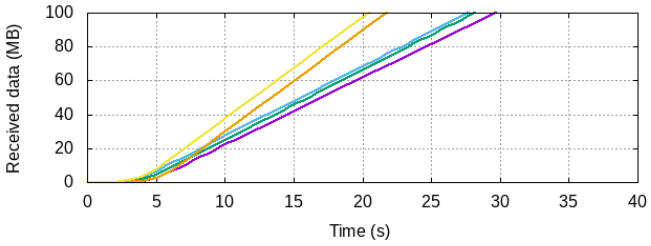

Fig. 7. Received data for 50 Mbps / 10 Mbps use case

B. Results for 50 Mbps forward without added losses

Figure 6 (resp. Figure 7) shows the received rate (resp. data) for the use case with 50 Mbps forward, 10 Mbps return and no added losses. The results show that splitting the TCP connections using PEP A and PEP B configurations does not improve much the performance compared to the solution without proxy. PEP C and PEP D have higher buffer sizes and this helps the connection in reaching the bottleneck capacity. Moreover, increasing the initial congestion window from 10 packets to 100 packets helps PEP D in getting up to speed faster by approximately 2 seconds. In the best possible configuration, TCP with PEP is thus able to deliver 2 MB in 3 seconds, 10 MB in 5 seconds and 100 MB in 20 seconds.

Looking at the received data evolution, we can contribute to the objectives definition of QUIC with the following values: 2 MB download in 3 seconds, 10 MB download in 5 seconds and 100 MB download in 20 seconds.

C. Results for 250 Mbps forward and 1% added random losses

Figure 8 (resp. Figure 9) shows the received rate (resp. data) for the use case with 250 Mbps forward, 3 Mbps return and 1 % random losses. The performance without proxy shows why the local recovery is essential for satellite-based systems.

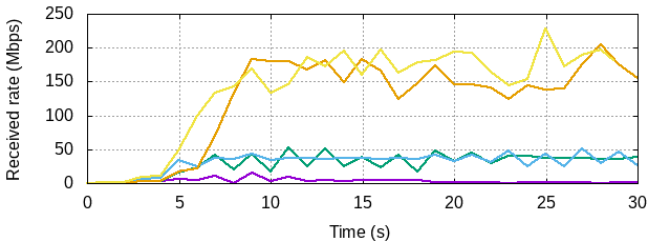

Fig. 8. Received rate for 250 Mbps / 3 Mbps use case (1 % random losses)

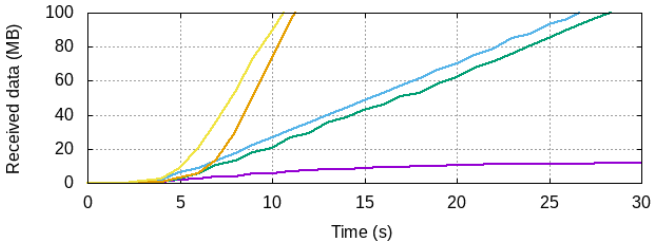

Fig. 9. Received data for 250 Mbps / 3 Mbps use case (1 % random losses)

This experiment has not been done multiple times to assess the relevance of this result. That being said, the trend is consistent with the important impact of losses on end-to-end TCP connections. Such as in Section IV-B, PEP A and PEP B cannot reach the available capacity even if they provide better performance when there is no TCP proxy. The results show that despite the local recovery and increased buffer sizes, PEP C and D cannot maintain a stable capacity. Looking at the received data evolution, we can contribute to the objectives definition of QUIC with the following values: 2 MB download in 3 seconds, 10 MB download in 6 seconds and 100 MB download in 10 seconds.

V. Dealing with this challenging context

Section IV proposed a framework that could be exploited to further define QUIC performance objectives or performance targets. This section describes some hints on how QUIC can be adapted to fulfill the targets that the community could propose. More information on these proposals can be found in [16].

Getting up to speed is one of the advantages of TCP proxies. This could be achieved within QUIC by adapting the congestion control and be more aggressive than the default Linux stacks [32]. Moreover, exploiting information of previous connections can help 0-RTT connections in fastly fetching the available capacity [33]. Fast and local recovery of losses are another advantages brought out by TCP proxy that may not be available if QUIC is exploited. There are activities in the IETF that may be relevant solutions for this issue, such as the MASQUE protocol [34] or introducing coding in QUIC [35]. Introducing QUIC proxies in satellite systems could help in tuning the protocol to the high BDP context [36]. Another issue that has been exhibited in this paper is the importance of increasing the buffer sizes on end points so that the available capacity can actually be exploited.

VI. Conclusion

QUIC is a transport layer revolution that may replace the historic and ossified TCP. Deploying transport level innovative concepts was complicated in the kernel space but kernel managers intended to prevent updates from collapsing the Internet [37]. Another important aspect of TCP design was its friendliness, its capability in keeping the Internet stable, and tunable to anybody's specific context. This is challenged with the deployment of QUIC. In this context, this paper analyses to what extent QUIC is an interesting opportunity for satellite broadband systems. Indeed, the reduced amount of handshakes helps in reducing the transmission time of short files. However, QUIC does not currently enable local recovery of losses; this can induce an important goodput reduction. As a result, QUIC may not outperform TCP proxy solutions but still achieves fairly good performance. Further acceptance of QUIC in satellite systems would be guaranteed with better performance in specific use-cases. We propose running code for an IETF document and based on an emulated platform and open-source software, this paper proposes example values as one piece of the puzzle. We encourage the community to reuse the framework and propose QUIC extensions able to challenge the current TCP proxy performances. An actual numbered performance objective requires consensus among the different actors. The objective should be challenging enough for satellite operators to allow QUIC traffic but reasonable enough to be deployable in the Internet.

As a future work, we plan on cross-testing different QUIC implementation in the scenarios that have been proposed in the IETF documents and evaluate the relevance of the proposed solutions to guarantee that targeted objectives are fulfilled and reduce the probability of satellite operators to block this traffic. Early results related to this future work has recently been presented at a IRTF PANRG interim meeting [38].

References

[1] "QUIC Working Group Website," sept 2018. [Online]. Available: https://quicwg.org
[2] J. Rüth *et al.*, "A First Look at QUIC in the Wild," in *Passive and Active Measurement Conference*, 2018, pp. 1–6.
[3] S. Iyengar and L. Niccolini, "Moving fast at scale, experience deploying ietf quic at facebook," https://conferences2.sigcomm.org/co-next/2018/slides/epiq-keynote.pdf, 2018.
[4] Jana Iyengar and Ian Swett, "QUIC Loss Detection and Congestion Control," IETF Secretariat, Internet-Draft draft-ietf-quic-recovery-27, March 2020.
[5] N. Cardwell *et al.*, "Bbr: Congestion-based congestion control," *Queue*, vol. 14, no. 5, pp. 50:20–50:53, Oct. 2016. [Online]. Available: http://doi.acm.org/10.1145/3012426.3022184
[6] Huitema, Christian and others, "Minimal implementation of the QUIC protocol," https://github.com/private-octopus/picoquic, 2020.
[7] "Quiche," https://quiche.googlesource.com/quiche/+/master/quic/, 2020.
[8] G. C. Projects, "Chromium source," sept 2018. [Online]. Available: https://chromium.googlesource.com/chromium/src.git
[9] J. Border *et al.*, "Performance enhancing proxies intended to mitigate link-related degradations," RFC 3135, June 2001.
[10] M. Marchese, "Tcp modifications over satellite channels: study and performance evaluation," *Int. J. Satellite Communications Networking*, vol. 19, pp. 93–110, 2001.
[11] C. Caini, R. Firrincieli, D. Lacamera, T. de Cola, M. Marchese, C. Marcondes, M. Sanadidi, and M. Gerla, "Analysis of tcp live experiments on a real geo satellite testbed," *Performance Evaluation*, vol. 66, no. 6, pp. 287 – 300, 2009. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0166531608001077
[12] A. Cardaci, L. Caviglione, E. Ferro, and A. Gotta, "Using spdy to improve web 2.0 over satellite links," *Int. J. Satell. Commun. Netw.*, vol. 35, no. 4, p. 307321, Jul. 2017. [Online]. Available: https://doi.org/10.1002/sat.1185
[13] Ludovic Thomas and others, "Google QUIC performance over a public SATCOM access," *International Journal of Satellite Communications and Networking*, vol. 37, no. 6, pp. 601–611, November 2019.
[14] J. Deutschmann, K. Hielscher, and R. German, "Satellite internet performance measurements," in *2019 International Conference on Networked Systems (NetSys)*, March 2019, pp. 1–4.

[15] R. Secchi *et al.*, "Performance analysis of next generation web access via satellite," *International Journal of Satellite Communications and Networking*, vol. 36, no. 1, pp. 29–43, dec 2016.

[16] Nicolas Kuhn et al., "QUIC for SATCOM," Internet-Draft draft-kuhn-quic-4-sat-04, March 2020.

[17] A. Langley *et al.*, "The quic transport protocol: Design and internet-scale deployment," in *SIGCOMM 17*, 2017.

[18] Z. Wang, "Navigation timing," W3C Recommendation, Dec. 2012.

[19] D. Mo, I. Swett, and N. Aviram, "Best practice to test congestion control part of quic," jan 2015. [Online]. Available: http://bit.ly/2MqnVpy

[20] "SeleniumHQ Browser Automation," https://www.selenium.dev/.

[21] Ryan Hamilton, "QUIC Discovery," The Chromium Projects, Design Document, Oct. 2014.

[22] Q. De Coninck *et al.*, "Pluginizing quic," in *SIGCOMM*, 2019.

[23] N. Duffield, A. Morton, and J. Sommers, "Loss episode metrics for ip performance metrics (ippm)," RFC 6534, May 2012.

[24] Ha, Sangtae and Rhee, Injong and Xu, Lisong, "CUBIC: a new TCP-friendly high-speed TCP variant," *ACM SIGOPS OSR*, vol. 42, no. 5, pp. 64–74, 2008.

[25] Y. Cheng *et al.*, "TCP Fast Open," IETF, RFC 7413, December 2014.

[26] Q. De Coninck *et al.*, "Multipath quic: Design and evaluation," in *CoNEXT*, 2017, pp. 160–166.

[27] F. Michel *et al.*, "QUIC-FEC: Bringing the benefits of Forward Erasure Correction to QUIC," in *IFIP Networking*. IEEE, 2019, pp. 1–9.

[28] Garrido Ortiz, Pablo and others, "rQUIC: Integrating FEC with QUIC for robust wireless communications," 2019.

[29] A. Ferrieux *et al.*, "Packet loss signaling for encrypted protocols," Internet-Draft draft-ferrieuxhamchaoui-quic-lossbits-03, January 2020.

[30] "OpenBACH : Network Metrology Testing Tool." [Online]. Available: http://www.openbach.org/

[31] C. Caini, R. Firrincieli, and D. Lacamera, "PEPsal: A Performance Enhancing Proxy for TCP Satellite Connections," *IEEE Aerospace and Electronic Systems Magazine*, vol. 22, no. 8, pp. B9–B16, Aug 2007.

[32] Y. Wang *et al.*, "Performance evaluation of quic with bbr in satellite internet," in *WiSEE*, 2018, pp. 195–199.

[33] Nicolas Kuhn et al., "Transport parameters for 0-RTT connections," Internet-Draft draft-kuhn-quic-0rtt-bdp-06, December 2019.

[34] David Schinazi, "The MASQUE Protocol," Internet-Draft draft-schinazi-masque-protocol-01, March 2020.

[35] Ian Swett and others, "Coding for QUIC," Internet-Draft draft-swett-nwcrg-coding-for-quic-04, March 2020.

[36] A. Abdelsalam and others, "QUIC-proxy based architecture for satellite communication to enhance a 5G scenario," in *ISNCC*, 2019, pp. 1–6.

[37] Christian Huitema, Brian Trammell, "A call for Congestion Defense in Depth," IETF 105 TSVAERA, 2019.

[38] Nicolas Kuhn, "Updates on QUIC Over In-sequence Paths with Different Characteristics," IETF PANRG Interim Meeting, 2020.