QUIC: Opportunities and threats in SATCOM

Nicolas Kuhn, François Michel, Ludovic Thomas, Emmanuel Dubois, Emmanuel Lochin, Francklin Simo, David Pradas

QUIC: Opportunities and threats in SATCOM

Nicolas Kuhn[1] | François Michel[2] | Ludovic Thomas[3] | Emmanuel Dubois[1] | Emmanuel Lochin[4] | Francklin Simo[5] | David Pradas[5]

[1]Navigation and Telecom, SATCOM System Team, CNES, Toulouse, France

[2]Institute of Information and Communication Technologies, Electronics and Applied Mathematics, UCLouvain, Ottignies-Louvain-la-Neuve, Belgium

[3]Département d'Ingéniérie des Systèmes Complexes, ISAE-SUPAERO, Université de Toulouse, France

[4]Networks and Communicating Systems Research Group, ENAC, Toulouse, France

[5]Viveris Technologies, Toulouse, France

Correspondence
Nicolas Kuhn, Navigation and Telecom, SATCOM System Team, Centre National d'Etudes Spatiales (CNES), 18, avenue Edouard Belin, 31400 Toulouse, France.
Email: nicolas.kuhn@cnes.fr

Summary

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 transmission time of short files. 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 were increased. Based on an emulated platform and on open-source software, this paper proposes values of performance metrics as one piece of the puzzle. The final performance objective requires consensus among the different actors. The objective should at least provide acceptable performance for satellite operators to allow QUIC traffic but reasonable enough to keep QUIC deployable on the Internet.

KEYWORDS
PEP, QUIC, SATCOM, TCP

1 | 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] For the latter, QUIC already accounts for 75% of their internet traffic.[4] 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. Indeed, 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 for the congestion control.[5] Even if not specified, newer mechanisms such as BBR[6] are undoubtedly already deployed in several QUIC implementations (picoquic,[7] quiche,[8] Chromium[9]). 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 (PEPs)[10] are currently doing with TCP. As a consequence, QUIC traffic will be seen as standard UDP traffic. This will not benefit from PEP optimization and could either be 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[11–13] 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[14,15] 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,[16] 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 Service Level Agreements (SLAs) 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 contribution identifying how to ensure acceptable protocol performance,[17] there is no actual evaluation framework available.

- This paper also proposes an evaluation framework, so that anyone can contribute to the performance objectives of QUIC.[18,19]
- This paper exploits the evaluation framework to assess the performance of `picoquic`[7] and `h2o/quicly`[20] server implementations over one scenario presented in an IETF contribution[17] with `picoquic` client.

The paper is organized as follows: in Section 2.1, 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 2.2, we also identify the End-to-End (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 2.3. In Section 3, based on the experimental results of Section 2, we propose a Strengths, Weaknesses, Opportunities, Threats (SWOT) analysis for QUIC over satellite from a satellite-operator point of view. In Section 4, 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. We evaluate `picoquic` server implementation with two different QUIC clients in Section 5. Section 6 describes some hints on how QUIC can be adapted to fulfill the objectives that the community could propose.

2 | 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 the original Google's version of QUIC with those of its TCP-proxy counterpart (in Section 2.1) and to (2) measure the E2E losses experienced without local recovery (in Section 2.2). To identify the impact of losses on an E2E connection, we assess the impact of random losses on a controlled satellite network (in Section 2.3).

2.1 | GQUIC performance on satellite public access

The rationale of this subsection is to compare the performance of Google QUIC (GQUIC) with those of its TCP-proxy counterpart to further identify issues that QUIC faces in satellite systems. While the results are related to the version of GQUIC that was deployed at the time of testing, they let us point the challenges that QUIC will face over a SATCOM access. 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.[21] 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 for wide comparison between GQUIC and TCP for web services since this depends on the characteristics of the web pages[16] and such comparison should rather consider the top-visited web page.[13] We rather aim at identifying trends in the results and the performance of the convergence of the congestion control when the full HTTP stack is considered. We focus on the page load time (PLT)[22] 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.

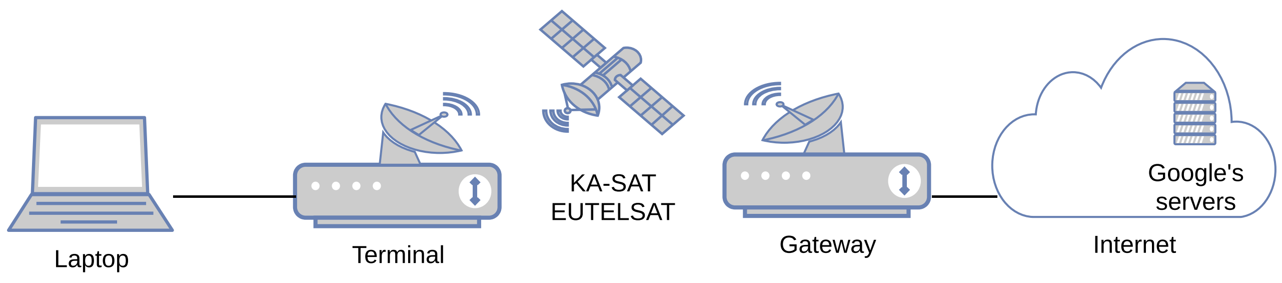

FIGURE 1 Testbed for GQUIC performance assessment

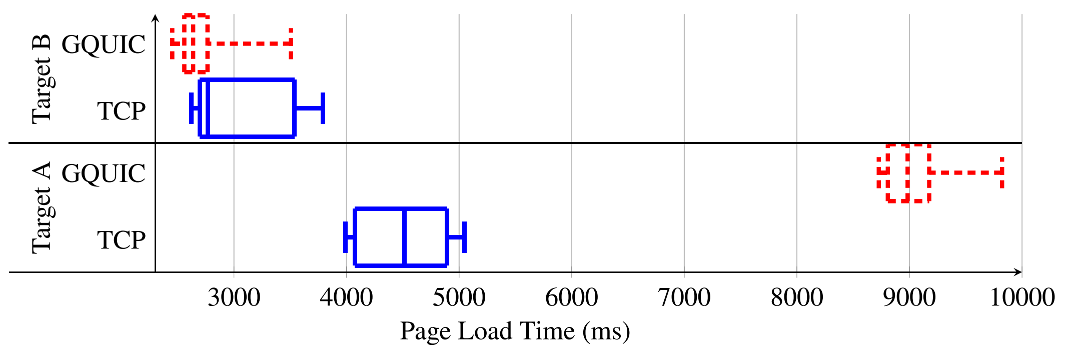

FIGURE 2 Distribution of the 5th, 25th, 50th, 75th, and 95th percentiles of the page load time (PLT) over 40 tests for Target A (one object of 5.3 MB) and for Target B (two object with a cumulated size of 11 kB)

Similarly, the setup of the experiments is as fair as possible for both protocols. In particular, Google has deployed the BBR congestion control:[6] We request therefore the same congestion control for GQUIC, using flags during the handshake[23] although we do not control the network stack of the remote server. Tests are performed using the Selenium automation tools,[24] and they have been designed based on Chrome behavior `67.0.3396.99`.[14,25]

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 the heavy target A. The HTTP GET messages using GQUIC have a completion 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 fewer 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 establishment is completed before the connection request actually reaches 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 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 s to probe the whole E2E link. On the contrary, TCP achieves a constant throughput in less than 50 ms. 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 a particular version of GQUIC that was deployed at the time of testing, 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.

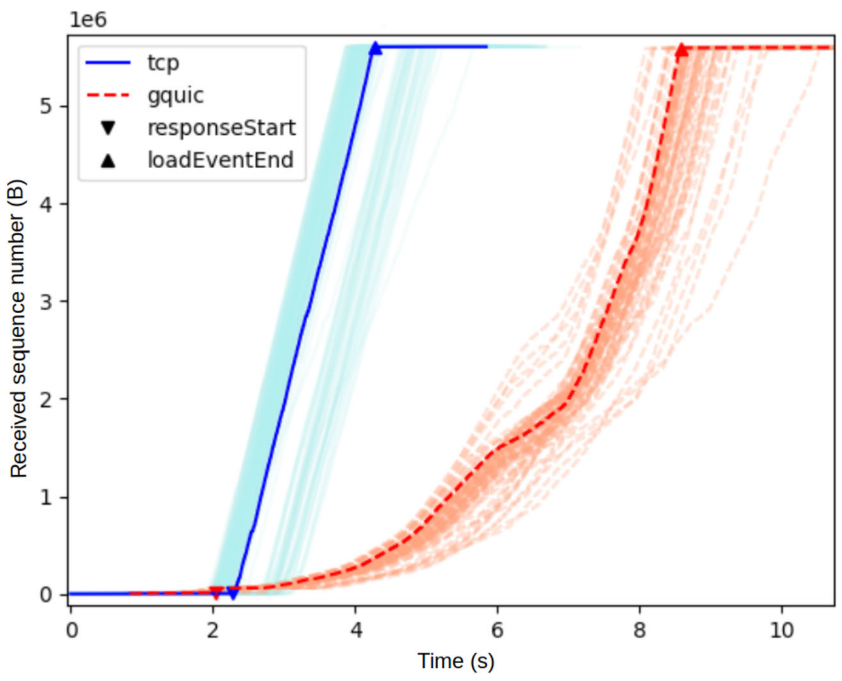

FIGURE 3 Sequence number evolution for target A. Focus on 2 out of 80 tests. Congestion control deployed at Google's servers is unknown but expected to be BBR. Chrome version `67.0.3396.99`

2.2 | Losses in public SATCOM access

In this section, we measure the end-to-end loss ratio that will be experienced without local recovery when using a public satellite broadband access. The testbed used is the same as in Figure 1 except that we now manage the server that is located at ISAE-SUPAERO.[1] The client laptop is still located at CNES.[2] The QUIC implementation is `picoquic`,[7] more specifically its PQUIC[26] fork. We study the losses experienced using satellite links under several conditions. Two experiments have been performed with different technologies 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, 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 1 shows the computed metrics. For each experiment, we report the uniform loss ratio ($loss_{uni}$), the simplified Gilbert model estimated transition probabilities, and the maximum burst size in packets (B_{max}). The simplified Gilbert model transition probabilities are expressed in the notation $P(s_1|s_2)$ proposed in RFC6534[27] where s_1 (resp. s_2) is the destination (resp. origin) state. The states are either the Good (g) or Bad (b) state. The uniform loss ratios and the simplified Gilbert model parameters are reported independently: (1) We measure the amount of packets that have been lost to compute a uniform loss ratio; and (2) we measure the amount of consecutive lost or received packets to estimate the Gilbert model parameters. In the Good state, all the packets are received, and in the Bad state, all the packets are lost. As shown on the table, the WiFi link seems to suffer from longer loss bursts and a higher uniform loss ratio. Due to the limited access on the satellite access, we could not drive an in-depth analysis on the relevance of using a Markov modelization for this system. That being said, they provide insights on the amount of losses and the approximate burst size 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.

2.3 | Impact of losses on an E2E connection

This section assesses the impact of end-to-end losses on a TCP connection over a real satellite platform. The rationale is to obtain base performances that QUIC should obtained in the same conditions. We denote by "QUIC targets" these performances objectives. 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.[28] Random

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

TABLE 1 End-to-end loss-related metrics from experiments with a wired and wireless access points

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

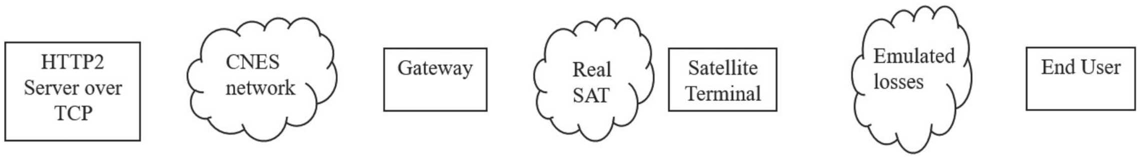

FIGURE 4 Testbed for impact of losses on E2E connection assessment

TABLE 2 Impact of losses on the goodput of a TCP connection

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

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 2. 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 even be higher, as shown in Section 2.2, these simple tests illustrate the impact 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.

3 | 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 over SATCOM systems.

3.1 | Strengths

We first discuss the strengths by 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 connection,[14] as shown in Section 2.1. Compared to the TCP + TLS1.3 combo, QUIC saves one RTT by doing both transport and cryptographic handshakes at once. The TCP Fast Open (TFO)[29] extension allows TCP to reach similar performance but still requires a TLS connection establishment. Moreover, TFO might not be enabled by default in the deployed TCP proxies. 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.

3.2 | 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 BDP of the link.

The slow congestion control convergence has a significant impact on the download time of large files as shown in Section 2.3. 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.

3.3 | 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 might 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[29] as only the endpoints are required to evolve. This would also reduce the price of satellite-operator networks, increase the resiliency of mobile scenarios, and improve the end-to-end security.

Another opportunity is the facility to extend QUIC. TCP only has a 40-byte header length to store extension information. QUIC is a frame-based protocol: such as defined in RFC9000, QUIC transmits frames that can carry application data (e.g., STREAM or DATAGRAME Frames) or control data (e.g., ACK Frames and CRYPTO Frames).[30] This allows to easily define new frames (Extension Frames) such as multipath,[31] Forward Erasure Correction,[32,33] or 0-RTT improvments[34] 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.

3.4 | 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 SLAs 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.[35]

3.5 | 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 when configuring 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.

4 | A FRAMEWORK FOR DEFINING OBJECTIVES FOR QUIC

Following the issues mentioned in Section 3.5, 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 contribution describes satellite accesses scenarios against which QUIC could be tested.[17] 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 code for the scenarios proposed in an IETF contribution[17] since the document only presents the configuration of the test and expected results.

We exploit the open-source tool OpenBACH, an open-source test orchestrator[36] that can 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 performance objectives require consensus among the different actors: The values proposed in this section are one piece of the puzzle. The objectives should at least provide acceptable performance for satellite operators to allow QUIC traffic but reasonable enough to be deployable in the Internet.

4.1 | 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.

The architecture depicts a distributed PEP configuration[10] where splitting is handle on both sides and ACK spoofing is enabled. PEPSal[37] operates a flow control like standard commercial PEPs. We will not dive into the full internal PEP mechanism implemented as this is both out of scope of this study and nonmandatory 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.

An IETF contribution proposes different scenarios and presents in details the amount of data generated, the required buffer sizes in bandwidth limiters and the output results for each scenario.[17] 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 3. We propose to extend the memory allocated to the connection and initial congestion window to help the congestion control to get up to speed. These 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.

4.2 | 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,

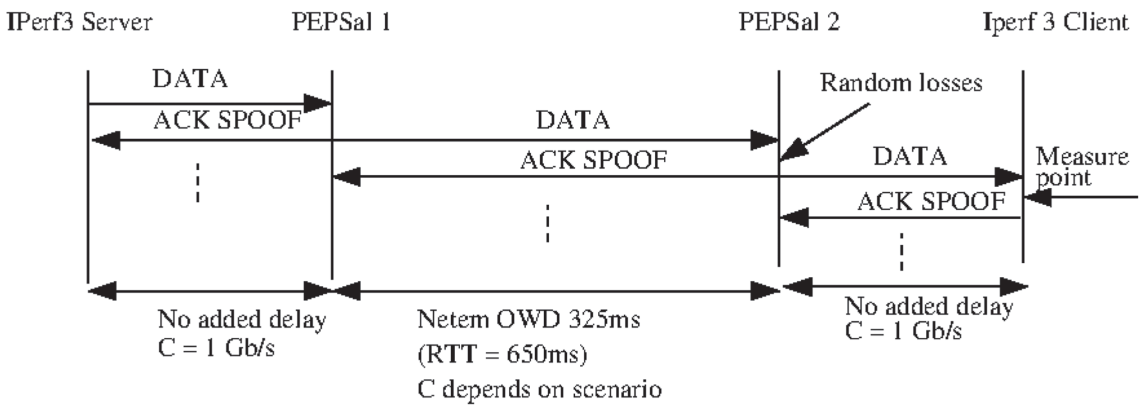

FIGURE 5 Testbed for QUIC objectives assessment

TABLE 3 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

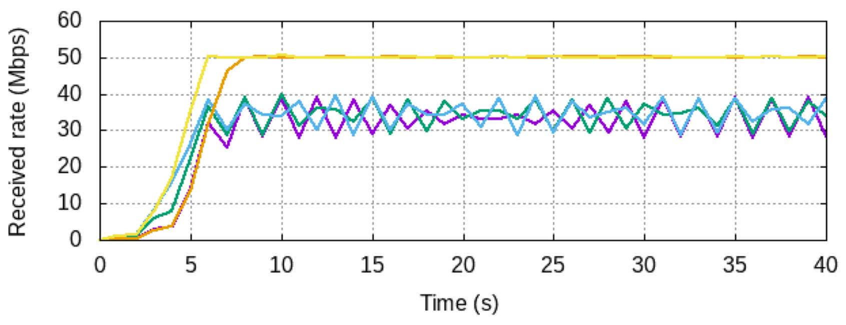

FIGURE 6 Received rate for 50 Mbps/10 Mbps use case—Legend is detailed in Table 3

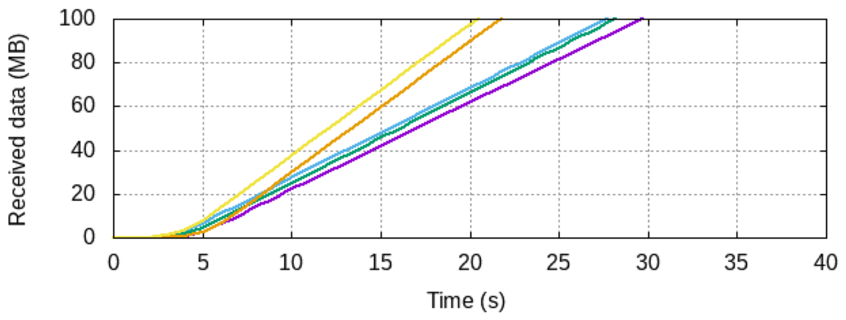

FIGURE 7 Received data for 50 Mbps/10 Mbps use case—Legend is detailed in Table 3

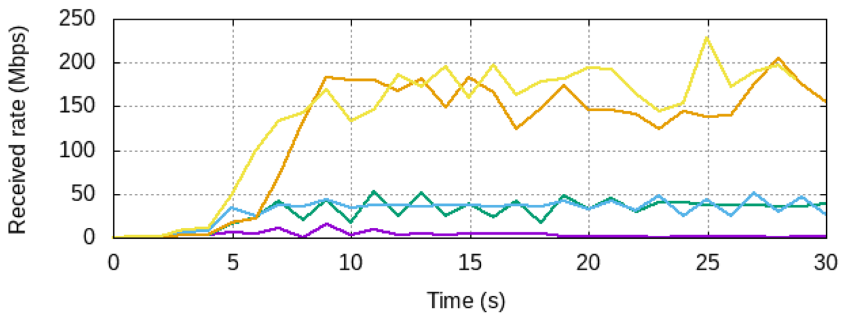

FIGURE 8 Received rate for 250 Mbps/3 Mbps use case (1 % random losses)—Legend is detailed in Table 3

increasing the initial congestion window from 10 packets to 100 packets helps PEP D in getting up to speed faster by approximately 2 s. In the best possible configuration, TCP with PEP is thus able to deliver 2 MB in 3 s, 10 MB in 5 s, and 100 MB in 20 s.

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

4.3 | 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. 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 4.2, 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 throughput. Looking at the received data evolution, we can contribute to the objectives definition of QUIC with the following values: 2 MB download in 3 s, 10 MB download in 6 s, and 100 MB download in 10 s.

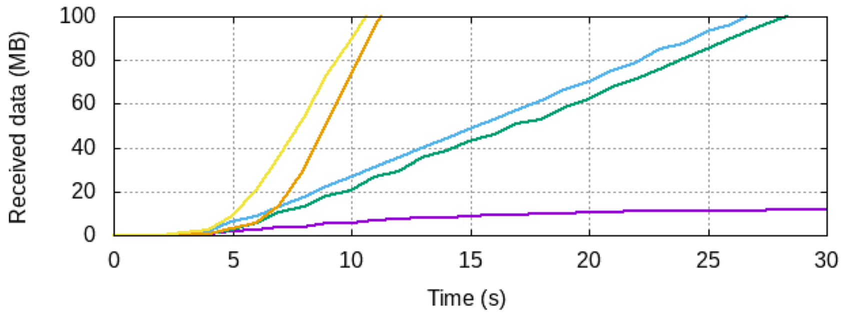

FIGURE 9 Received data for 250 Mbps/3 Mbps use case (1 % random losses)—Legend is detailed in Table 3

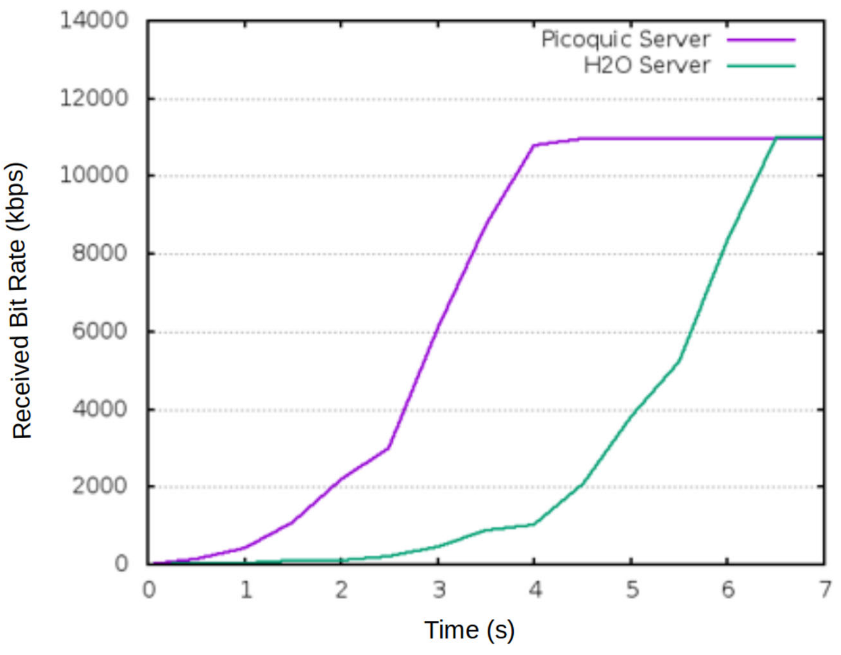

FIGURE 10 Received data as a function of time for a `picoquic` client for a 10-MB file

5 | COMPARING QUIC SERVERS USING THE EVALUATION FRAMEWORK

This section exploits the open-source scripts, parts of the evaluation framework, to assess the performance of `picoquic`[7] and `h2o/quicly`[20] server implementations serving a `picoquic` client over the scenarios presented in an IETF contribution.[17] The results presented in this section can be directly reproduced using OpenBACH[18] and adapted using one of the OpenBACH examples.[19]

5.1 | Testbed description

The testbed is composed of a QUIC server and a QUIC client, interconnected via a middlebox. The bandwidth emulator uses `tc` to emulate delay, bandwidth, and loss on both forward and return links. For `picoquic` client and server and for `h2o/quicly`, the version used is compliant with `h3-25`. The tests have been executed in early 2020 so these results might differ from up-to-date stacks.

This paper presents the results for the 50 Mbps forward without added losses, `picoquic` client, and server and `h2o/quicly` server.

5.2 | `picoquic` and `h2o/quicly` servers comparison over the 50-Mbps scenario

Figure 10 (resp. Figure 11) shows the received data (resp. received data rate) as a function of time for a `picoquic` client for a 10-MB file. We recall that the emulation with PEP and TCP was able to deliver 2 MB in 3 s and 10 MB in 5 s (see Section 4.2).

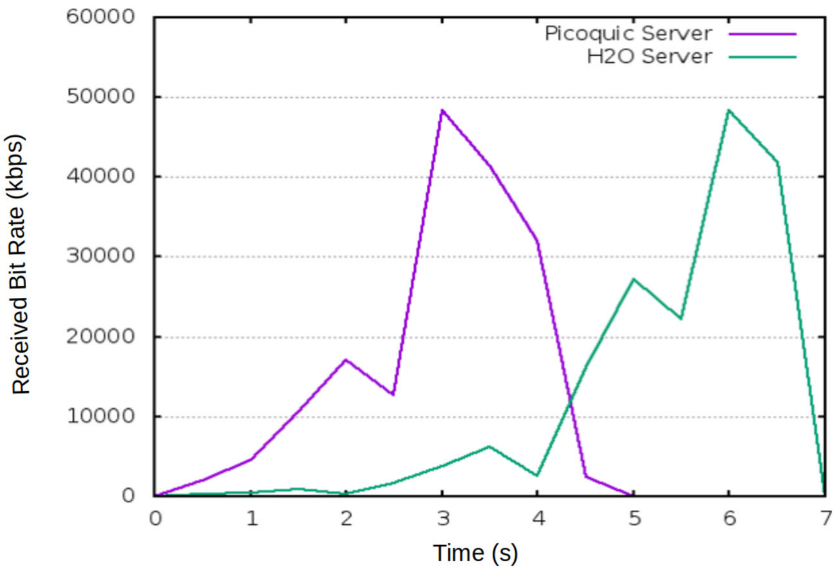

FIGURE 11 Received data rate as a function of time for a `picoquic` client for a 10-MB file

Figure 10 shows that `picoquic` server achieves similar performance as those obtained with TCP and PEP. `h2o/quicly` achieves good performance that are slightly below those of `picoquic` and the target. Looking at Figure 11, we can see that the two implementations show the same trend in data rate evolution. That being said, `picoquic` seems to start the transmission of data earlier, which explains the gains in data transmission time.

The difference in performance may be the result of many different parameters that may be more or less adequate for a satellite scenario. The initial congestion window could be tuned to decrease the convergence time of the congestion control, but these values were close (10 $\times$ 1140 bytes for `picoquic` and 10 $\times$ 1280 for `h2o/quicly`). However, the *INITIAL_RTT* was 250 ms (resp. 100 ms), the *ACK_MAX_DELAY* 10 ms (resp. 25 ms), and the congestion control BBR (resp. Reno) for `picoquic` (resp. `h2o/quicly`). Adequate tuning for satellite scenarios may be needed to reach performance comparable to those with TCP and PEP.

5.3 | Comments on the 250-Mbps scenario

Results for the 250-Mbps scenario were presented in the Path Aware Networking Research Group (PANRG) IETF105 meeting.[38] These results are available online. In this scenario, the only cases where the targeted performance was reached are with a `picoquic` client and server combination. No other combination could reach them. This confirms the importance of not only working at the server but also at the client for performance improvement.

6 | DEALING WITH THIS CHALLENGING CONTEXT

Section 4 proposed a framework that could be exploited to further define QUIC performance objectives or performance targets. Section 5 confirms that current implementations of QUIC can achieve quite good performance in some satellite use cases. However, reaching the bottleneck capacity for higher data rates or when there are transmission losses is more challenging.

This section describes some hints on how QUIC can be adapted to fulfill the targets that the community could propose. Further details can be found in an IETF contribution.[17]

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.[39] Moreover, exploiting the information of previous connections can help 0-RTT connections in fastly fetching the available capacity.[34] Fast and local recovery of losses is another advantage of TCP proxy that might not be available if QUIC is exploited. There are activities in the IETF that may be relevant for this issue, such as the MASQUE protocol[40] or introducing coding in QUIC.[41] Introducing QUIC proxies in satellite systems could help in tuning the protocol to the high BDP context.[42] The introduction of QUIC proxies would require to split the TLS connexion and may then be relevant for enterprise networks. Otherwise, this would require the approval of the client and may encounter deployment issues. Another issue that has been exhibited in this paper is the importance of increasing the buffer sizes on endpoints to better exploit the available capacity.

7 | 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.[43] Another important aspect of TCP design was 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. 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. In this paper, we have cross-tested different QUIC implementation in the scenarios that have been proposed in the IETF contribution and evaluate the relevance of the proposed solutions.

A fully specified performance objective requires consensus among the different actors. The objective should at least provide acceptable performance for satellite operators to allow QUIC traffic but reasonable enough to be deployable in the Internet.

CONFLICT OF INTEREST

The authors declare that they have no conflicts of interest.

DATA AVAILABILITY STATEMENT

The links are available on https://forge.net4sat.org or on authors personal GITHUB pages, https://github.com/NicoKos/.

ORCID

Nicolas Kuhn https://orcid.org/0000-0001-6671-3490
Emmanuel Lochin https://orcid.org/0000-0002-6305-2188

REFERENCES

1. IETFQUIC Working Group. Quic working group website *Edited by* IETF. https://quicwg.org; 2018.
2. Rüth J, Poese I, Dietzel C, Hohlfeld O. A First Look at QUIC in the Wild. In: Passive and Active Measurement Conference. Springer; 2018; Cham: 255-268.
3. Iyengar S. Moving fast at scale: Experience deploying ietf quic at facebook. In: Proceedings of the Workshop on the Evolution, Performance, and Interoperability of QUIC; 2018: Keynote. https://doi.org/10.1145/3284850.3322434
4. How Facebook is bringing QUIC to billions. https://engineering.fb.com/2020/10/21/networking-traffic/how-facebook-is-bringing-quic-to-billions/; 2020.
5. Iyengar J, Swett I. QUIC Loss Detection and Congestion Control. *RFC*. 9002, Internet Requests for Comments; 2021.
6. Cardwell N, Cheng Y, Gunn CS, Yeganeh SH, Jacobson V. Bbr: Congestion-based congestion control. *ACM Queue*. 2016;14(5):50:20-50:53. https://doi.org/10.1145/3012426.3022184
7. Huitema C. Minimal implementation of the QUIC protocol. GitHub. https://github.com/private-octopus/picoquic; 2020.
8. Quiche. https://quiche.googlesource.com/quiche/+/master/quic/; 2020.
9. Google Chromium Projects. Chromium source *Edited by* Google. Website, https://chromium.googlesource.com/chromium/src.git; 2018.
10. Border J, et al. Performance enhancing proxies intended to mitigate link-related degradations. RFC 3135; 2001.
11. Marchese M. TCP modifications over satellite channels: study and performance evaluation. *Int J Satell Commun*. 2001;19(1):93-110. https://doi.org/10.1002/sat.682
12. Caini C, Firrincieli R, Lacamera D, et al. Analysis of tcp live experiments on a real geo satellite testbed. *Perform Eval*. 2009;66(6):287-300.
13. Cardaci A, Caviglione L, Ferro E, Gotta A. Using spdy to improve web 2.0 over satellite links. *Int J Satell Commun Netw*. 2017;35(4):307-321. https://doi.org/10.1002/sat.1185
14. Thomas L, Dubois E, Kuhn N, Lochin E. Google QUIC performance over a public SATCOM access. *Int J Satell Commun Netw*. 2019;37(6):601-611.
15. Deutschmann J, Hielscher K, German R. Satellite internet performance measurements. In: 2019 International Conference on Networked Systems (NetSys); 2019:1-4.
16. Secchi R, Mohideen AC, Fairhurst G. Performance analysis of next generation web access via satellite. *Int J Satell Commun Netw*. 2016;36(1):29-43. https://doi.org/10.1002/sat.1201
17. Jones T, Fairhurst G, Kuhn N, Border J, Emile S. Enhancing Transport Protocols over Satellite Networks. *Internet-Draft - Work-in-Progress*. draft-jones-tsvwg-transport-for-satellite-00, IETF Secretariat; 2021. Working Draft.
18. CNES, ed.. Example of scenarios composition to cross-test QUIC implementations. https://forge.net4sat.org/openbach/openbach-extra/blob/master/executors/examples/quic_configure_link.py
19. CNES, ed.. OpenBACH User Manual. https://wiki.net4sat.org/doku.php?id=openbach:manuals:2.x:user_manual:index
20. Oku K, et al. A modular QUIC stack designed primarily for H2O. GitHub, https://github.com/h2o/quicly; 2020.
21. Langley A, Riddoch A, Wilk A, 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, SIGCOMM '17; 2017:183-196. https://doi.org/10.1145/3098822.3098842
22. Wang Z. Navigation timing. W3C Recommendation. http://www.w3.org/TR/2012/REC-navigation-timing-20121217/; 2012.

23. Mo D, Swett I, Aviram N. Best practice to test congestion control part of quic. Forum discussion. http://bit.ly/2MqnVpy; 2015.
24. SELENIUM.
25. Hamilton R. Quic discovery. Design Document, The Chromium Projects. https://www.chromium.org/quic; 2014.
26. De Coninck Q, Michel F, Piraux M, et al. Proceedings of the ACM Special Interest Group on Data Communication, SIGCOMM '19, 59 –74; 2019.
27. Duffield N, Morton A, Sommers J. Loss episode metrics for ip performance metrics (ippm). *RFC*. 6534; 2012.
28. Ha S, Rhee I, Xu L. Cubic: A new tcp-friendly high-speed tcp variant. *SIGOPS Oper Syst Rev*. 2008;42(5):64-74.
29. Cheng Y, Chu J, Radhakrishnan S, Jain A. TCP Fast Open. *RFC*. 7413, IETF; 2014.
30. Iyengar J, Thomson M. QUIC: A UDP-Based Multiplexed and Secure Transport. *RFC*. 9000, RFC Editor; 2021. Internet Requests for Comments.
31. De Coninck Q, Bonaventure O. Multipath quic: Design and evaluation. In: Proceedings of the 13th International Conference on Emerging Networking EXperiments and Technologies, CoNEXT '17; 2017. https://doi.org/10.1145/3143361.3143370
32. Michel F, De Coninck Q, Bonaventure O. Quic-fec: Bringing the benefits of forward erasure correction to quic. In: 2019 IFIP Networking Conference (IFIP Networking); 2019:1-9.
33. Garrido P, Sanchez I, Ferlin S, Aguero R, Alay O. rquic: Integrating fec with quic for robust wireless communications. In: 2019 IEEE Global Communications Conference (GLOBECOM); 2019:1-7.
34. Nicolas Kuhn ES. Transport parameters for 0-RTT connections. *Internet-Draft - Work-in-Progress*. draft-kuhn-quic-0rtt-bdp-08; 2019.
35. Ferrieux A, et al. Packet loss signaling for encrypted protocols. *Internet-Draft Work-in-Progress*. draft-ferrieuxhamchaoui-quic-lossbits-03; 2020.
36. CNES, ed.. OpenBACH : Network Metrology Testing Tool. http://www.openbach.org/
37. Caini C, Firrincieli R, Lacamera D. PEPsal: a performance enhancing proxy for TCP satellite connections. *IEEE Aerosp Electron Syst Mag*. 2007;22(8): B9-B16.
38. Kuhn N. Updates on QUIC Over In-sequence Paths with Different Characteristics. *PANRG Interim Meeting*; 2020.
39. Wang Y, Zhao K, Li W, Fraire J, Sun Z, Fang Y. Performance evaluation of quic with bbr in satellite internet. In: 2018 6th IEEE International Conference on Wireless for Space and Extreme Environments (WiSEE); 2018:195-199.
40. Schinazi D. The MASQUE Protocol. *Internet-Draft - Work-in-Progress*. draft-schinazi-masque-protocol-01; 2020.
41. Swett I, et al. Coding for QUIC. *Internet-Draft - Work-in-Progress*. draft-swett-nwcrg-coding-for-quic-04; 2020.
42. Abdelsalam A, Luglio M, Quadrini M, Roseti C, Zampognaro F. Quic-proxy based architecture for satellite communication to enhance a 5g scenario. In: 2019 International Symposium on Networks, Computers and Communications (ISNCC); 2019:1-6.
43. Christian Huitema BT. A call for Congestion Defense in Depth. *IETF 105 TSVAERA*; 2019.

AUTHOR BIOGRAPHIES

Nicolas Kuhn obtained his Masters from ISAE - ENSICA in aeronautical engineering in 2010. In 2013, he obtained a doctorate from ISAE and NICTA. From January 2014 to September 2015, he was a postdoctoral fellow at the Institut Mines-Télécom (Télécom Bretagne) and particularly involved in the European RITE project. Since September 2015, he has been working at the Center National d'Etudes Spatiales (CNES) as a research engineer. His research focuses on transport layer issues in space telecommunications and how end-to-end service can be achieved in this challenging environment. Thus, it also works on the quality of experience, quality of service, access methods and cross-layer designs. He is particularly involved in the standardization of the QUIC protocol at the IETF and the need for protocol adaptation for satellite communications.

François Michel is a PhD student at UC Louvain, under the supervision of Olivier Bonaventure. His research interest mostly resides in network coding and transport protocol performance and their extension mechanisms.

Ludovic Thomas obtained his Master's degree fin Aerospace Engineering, Planetary science and Astrophysics from ISAE - SUPAERO in 2018. In May 2018, he was an intern at CNES where he was researching issues and solutions in the transport-layer space to power Internet through geostationary access links. Since 2018, he is a PhD student at ISAE - SUPAERO and EPFL. He has been focusing on performance analysis of Time-Sensitive Networking (TSN) technologies using the Network Calculus framework.

Emmanuel Dubois is a Network and Telecom research engineer. He joined Centre National d'Etudes Spatiales in Toulouse in 2008. He holds a PhD dealing with convergence in satellite networks. His main topics of interest include therefore network architecture convergence, cross-layering optimization, and also transport layer, QoS, and resource management. He has been working on several tools for research and development in satcom topics from an open source satellite emulation platform called OpenSAND to a multipurpose metrology tool called OpenBACH.

Prof. **Emmanuel Lochin** received his PhD from the LIP6 laboratory of Pierre and Marie Curie University - Paris VI in December 2004 and the Habilitation Thesis (Habilitation à Diriger des Recherches) in October 2011 from Institut National Polytechnique de Toulouse (INPT). From July 2005 to August 2007, he held a researcher position in the Networks and Pervasive Computing research program at National ICT Australia, Sydney. He then held a full professor position at ISAE-SUPAERO from September 2007 to March 2020 and has co-founded SPEERYT in July 2018 to stimulate the development and diffusion of an on-the-fly coding scheme named Tetrys. Before SPEERYT, this technology was transferred by TTT to a world leader in Internet content distribution. He is now full professor at ENAC since April 2020 and is also member of TéSA laboratory and computer networking expert in the TeSA scientific committee.

Francklin Simo received the MSc degree in computer science from Sorbonne University Paris Nord in 2014 and the PhD degree in computer networking from National Institute of Applied Science (INSA) of Toulouse in 2018. He participated in the French National Research Project - Application Driven Networking where his research was on aspects related to programmable networks involving NV, NFV and SDN. Since July 2018, he has been working at Viveris Technology as a network engineer where he contributes as a developer to open source projects, namely, OpenBACH and OpenSAND and also participates in various R&T and several European projects. His current research interests focus on networking via satellite and particularly on performance issues experienced by services and applications.

David Pradas obtained his MSc degree in telecommunications engineering from UPC in 2006, and he owns a PhD (2011) from ISAE-SUPAERO and from UAB, co-funded by CNES and UAB. During his PhD, he focused on cross-layer designs and methodologies for satellite broadband networking and participated in several international research projects (MOVISAT, SatNEx I & II). He is now the technical manager of the network/telecom R&D team at Viveris Technologies. In the frame of his activity in Viveris (since 2012), he has contributed to the OpenSAND satellite emulator and the OpenBACH metrology tool and has led more than 20 R&T (Research & Technology) projects for the CNES. His current research interests are the evaluation of satcom services in terms of QoS/QoE with a focus on the optimization of higher layers in satellite and hybrid systems.