QUIC Congestion Control Algorithm Characteristics in Mixed Satellite–Terrestrial Emergency Communication Scenarios

Armir Bujari[a], Mirko Franco[b], Claudio E. Palazzi[b], Mattia Quadrini[c], Cesare Roseti[d], Francesco Zampognaro[e,1]

[a]*Department of Computer Science and Engineering, University of Bologna, Bologna, Italy*
[b]*Department of Mathematics, University of Padua, Padua, Italy*
[c]*Research and Development, RomARS S.r.l., Rome, Italy*
[d]*Electronics Engineering Dpt., University of Rome "Tor Vergata", Rome, Italy*
[e]*Department of Human Sciences, Link Campus University, Rome, Italy*

Abstract

Reliable communications play a pivotal role in ensuring an efficient response and the coordination of recovery and rescue efforts. However, conventional communication methods may not always be accessible or dependable in such situations. In such circumstances, constellations of Low Earth Orbit (LEO) satellites can provide high bandwidth capabilities with relatively low latency, making them well-suited for supporting on-the-ground disaster management teams. Satellites can either complement or replace terrestrial telecommunication infrastructures. In this context, reliance on the recently defined QUIC protocol allows for a seamless transition from terrestrial to satellite communication as needed. Therefore, we investigate the possible use of a dual-stack node architecture along with the employment of the QUIC transport protocol for emergency communications, assuming that the backhaul link used to transfer users' applications data may need to be changed (seamlessly). We conduct an extensive emulation study, evaluating the performance of QUIC under varying queuing policies and Congestion Control Algorithm (CCA) behavior, providing practical insights and recommendations to enhance the protocol's efficiency and robustness. The key aspects and configurations of QUIC protocol stack are identified, presenting optimal communication configurations leveraging CoDel and BBR CCA.

Keywords:
QUIC, LEO constellation, CCA, emergency communications, network queues

1. Introduction

Telecommunication networks play a crucial role in establishing dependable, swift, and resilient communication channels among individuals, businesses, and institutions globally. This involves not only ensuring the speed and reliability of communication but also safeguarding the security and integrity of both the communication processes and the underlying network infrastructure. However, in areas struck by disasters or during emergencies in general, the reliability of telecommunication infrastructures is often uncertain. In these contexts, communication is paramount for coordinating both military and civilian personnel, facilitating data exchange, and enhancing operators' situational awareness [1]. This holds not only for the well-being of individuals but also for the protection of valuable assets [2].

Currently, emergency communications heavily rely on digital applications and services, facilitating voice, multimedia, and data transfers through IP-based, wireless access networks [3]. This setting ensures mobility and optimal performance, particularly with handheld/smart devices. At the same time, the volume of data exchanged is steadily increasing, especially with the deployment of advanced applications like augmented reality, high-resolution aerial photography (via drones), and the management of extensive datasets. Additionally, in-network services and the implementation of Multi-Access Edge Computing (MEC) are commonly employed to provide localized, low-latency value-added services without necessitating communication with the core-backed network services [4, 5]. In these scenarios, satellite communications (SatCom) can assume a pivotal role. Within the SatCom coverage area, digital communication services can be directly activated via ground terminals, independent of the condition of ground infrastructures in the designated area. Consequently, the utilization of terrestrial mobile networks, when accessible, can be seamlessly transitioned to SatCom (and vice versa), ensuring service continuity against potential terrestrial network disruptions. Therefore, the establishment of an intelligent, local node capable of overseeing terrestrial and satellite network access, as well as managing the local distribution of traffic and the execution of local/virtual services, becomes imperative [6].

In this work, we consider an emergency scenario where the backhaul link used to transfer users' applications data needs to be changed seamlessly. In this context, we explore the possible use of a dual-stack (SatCom and Mobile) node architec-

Email addresses: armir.bujari@unibo.it (Armir Bujari), mifranco@math.unipd.it (Mirko Franco), cpalazzi@math.unipd.it (Claudio E. Palazzi), quadrini@romars.tech (Mattia Quadrini), roseti@ing.uniroma2.it (Cesare Roseti), f.zampognaro@unilink.it (Francesco Zampognaro)

[1]Authors in alphabetical order

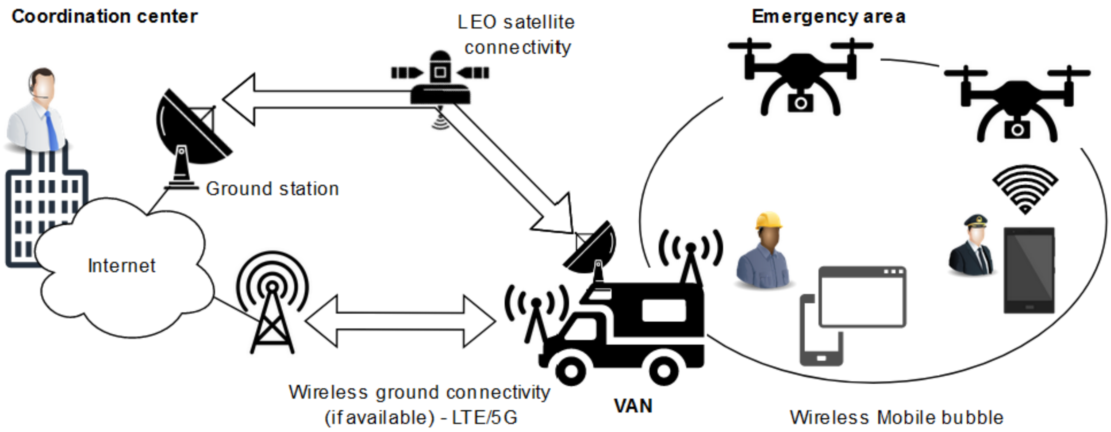


Figure 1: High level system architecture for data transfer in emergency scenarios.

ture utilizing the QUIC protocol [7], a recently formulated reliable transport protocol operating atop UDP and allowing opportunistic, efficient and seamless transitions among different kinds of communication means. QUIC incorporates and improves upon several fundamental principles of TCP, positioning itself as a more robust and feature-rich alternative to the conventional TCP. Notably, QUIC constitutes a crucial component of the widely adopted HTTP/3 specification [8]. Of particular interest is the aforementioned inherent path-migration feature, enabling the continuity of end-to-end data sessions even amid changes in access networks and IP addresses, a capability not shared by TCP.

A key aspect to achieve effective resource utilization and to ensure applications Quality of Service (QoS) in challenging communication scenarios, as the one addressed in this work, is the Congestion Control Algorithm (CCA) governing the QUIC connection's data transfer. Consequently, in the present work, we conduct an extensive emulation study campaign based on a reference Open Source QUIC stack, namely ngtcp2[2]. This work extends the previous work by the same authors [6] by moving the analysis from simulated QUIC to real implementations, adding more challenging network configurations with smart queues and alternative CCA to the default NewReno one.

In particular, the extended testing and realistic configurations implemented in the present work, apart from confirming QUIC robustness to link change, allowed us to identify the key CCA characteristics to ensure an efficient data transfer, as close as possible to the available link capacity but not reducing applications interactivity. As it is extensively discussed in the second part of the paper, the key findings of our work are related to the identification of suitable network configurations, in particular associated with smart queues and state-of-the-art CCA, which is BBR [9].

It is worth to remark that the proposed solution lends itself to additional enhancements, such as the joint exploitation of terrestrial and SatCom by employing multipath extensions of the QUIC protocol stack, which is currently in the latest stages of standardization [10].

[2]`https://nghttp2.org/ngtcp2/index.html`

The article is organized as follows: Section 2 discusses the envisaged reference architecture, its design principles and capabilities. Section 3 delves into the details of the QUIC stack, its features and functionalities which we leverage in our solution. In Section 4 we overview related work on the use of QUIC over satellite links. Section 5 introduces the emulation environment and the representative configuration settings, while Section 7 discusses the various identified tradeoffs. Finally, in Section 8 we draw our conclusions.

2. Reference Architecture

In the current communication networks landscape, commercially accessible constellations of Low Earth Orbit (LEO) satellites, have gained prominence due to their high satellite count, ranging from hundreds to thousands, providing pervasive coverage and communication opportunities. A notable example is Starlink [11], which has seen recent utilization in supporting communication in conflict areas. However, it is essential to note that achieving broadband connectivity with LEO constellations requires a terminal antenna that is not compact enough to be seamlessly integrated into devices of smartphone size. Additionally, as previously discussed, the deployment of a MEC node within the targeted area is crucial. This MEC node is designed to execute localized operations on traffic, encompassing services such as data mining and fusion, and to offer advanced network services such as Mission Critical Push to Talk, centralized data storage, and more.

To fulfil these essential service requirements, we propose an emergency communication architecture outlined in Figure 1, wherein the SatCom antenna is positioned within a larger mobile unit, such as a Vehicle Area Network (VAN). This backhaul connectivity guarantees a dependable and swiftly deployable communication link with a coordination centre positioned beyond the emergency zone. Additionally, thanks to the presence of a dedicated access node, we can explore the utilization of mobile terrestrial networks (LTE/5G/...), either in combination with or as an alternative to SatCom, potentially offering superior performance. The VAN can establish a multi-access setup via a dual-stack node, dynamically selecting the optimal backhaul link based on prevailing circumstances. It can seamlessly hand over existing connections from one link (considering the terrestrial option, if available, as the preferred choice) to another link (e.g., the LEO satellite) when necessary. Subsequently, the VAN extends connectivity to end-users through standard Wi-Fi or licensed standalone local LTE or 5G bubbles, denoted in the figure as the *Wireless Mobile bubble*. Finally, the VAN serves as the host for all local services and equipment required for MEC. Clearly, the number of end-hosts (hence even the number of flows) can vary depending on the scenario and the system results as scalable as the chosen backhaul connectivity option allows.

In this scenario, we explore a situation wherein a large data file needs to be transmitted from an operator device located within an emergency area (or within one of the emergency areas simultaneously present) to the remotely located coordination centre. This file may be a high resolution image or a video providing detailed information of the emergency (e.g., an accident, an earthquake, a flood, a collapsed bridge, ...) used to coordinate the operations. To this aim a high throughput, a low delay, and a resilient connectivity are crucial. Our dual-stack architecture employing QUIC provides the possibility to seamlessly switch from terrestrial to satellite communication (or between any two implemented communication technologies) when the former is failing due to the emergency itself and to implement a specific congestion control mechanism to improve the network performance considering the specific features of the different connectivity means in terms of bandwidth, delay, errors, and concurrent traffic.

Clearly, the considered scenario is just an example of the many possible employments, which may include smart city management and precision agriculture [12, 13]. In any case, it is worth noting that the backhaul connectivity options could be generally managed by different Network Operators, a Mobile Network Operator (MNO) and a Satellite Network Operator (SNO), with different network addressing, NAT configurations and connection characteristics. The transfer is performed using the QUIC protocol, in the uplink direction, and may suffer from link change (outage) during its establishment. Our analysis revolves around the interplay of network and QUIC stack mechanisms, leaving the study of additional local added-value services, e.g., joint exploitation of terrestrial-SatCom networks, as a future work.

3. The QUIC protocol

QUIC is described as a versatile, secure, and connection-oriented transport protocol for the Internet, built upon UDP. It inherits the benefits of TCP, such as reliable data delivery and congestion control [7]. Additionally, QUIC streamlines connection establishment by simultaneously negotiating cryptographic (TLSv1.3) and transport parameters, facilitating swift data exchange, even during initial phases, through its 0-RTT handshake feature.

The identification of QUIC connections is facilitated by a Connection Identifier (CID), a 64-bit unsigned number that is randomly generated by the server. Each QUIC connection encompasses multiple Streams, which represent ordered sequences of bytes. These streams can serve either unidirectional purposes, catering to live media streams, or bidirectional functionalities, which are well-suited for handling HTTP requests and responses. Moreover, QUIC packets are designed to accommodate one or multiple frames, effectively multiplexing data from various streams. This multiplexing process is guided by priority information provided by the endpoints, ensuring efficient resource utilization. Within the header of QUIC packets, the frame type field delineates the nature of the transmitted data, distinguishing between user data and signaling information such as Acknowledgments (ACKs).

QUIC has swiftly emerged as the preferred transport protocol for HTTP/3, offering unparalleled flexibility and performance enhancements when compared to HTTP/2 over TCP. The seamless integration of QUIC with HTTP/3 has revolutionized web communication, facilitating faster and more reliable data transfer over the Internet. This transition has been driven by QUIC's innate ability to adapt to dynamic network conditions while delivering superior performance and enhanced security features. As a result, QUIC has become synonymous with modern web communication standards, empowering organizations to deliver optimized web experiences to their users [8].

3.1. Congestion Control and Loss Detection

The QUIC working group at IETF suggested utilizing the comprehensive expertise of TCP congestion control by transferring its fundamental operational principles to QUIC [14]. This document specifies that the primary or mandatory sender-side Congestion Control Algorithm (CCA) is practically equivalent to TCP NewReno's CCA, except for some minor modifications [15]. Nonetheless, QUIC strives to implement a flexible congestion control mechanism, enabling users to choose from various CCAs. This approach allows for the adoption of alternative TCP CCA variants that provide tailored optimizations for specific environments. Examples are TCP Westwood [16], suitable to wireless links, TCP Hybla [17], TCP Lybra [18], and Vegas [19], showing some benefits in satellite scenarios, up to the recent TCP Cubic [20] and BBR [9], whose goal is to be effective in a large set of configurations. Implementers are free to explore various CCA variations in QUIC, resulting in different QUIC protocol stacks offering distinct CCAs by default.

While QUIC inherits the fundamental congestion control logic from TCP, it undergoes enhancements to suit its unique communication environment. These improvements include the integration of additional loss-recovery mechanisms, such as Forward-RTO and Early Retransmit. Forward-RTO effectively manages spurious timeouts, while Early Retransmit expedites retransmission in scenarios with small windows, employing a reduced number of duplicated ACKs as loss indicators. Consequently, QUIC obviates the necessity for Fast-Retransmit and Fast-Recovery mechanisms [21].

Moreover, QUIC offers more granular feedback information for loss detection. It employs a monotonically increasing packet number for both original and retransmitted packets, ensuring clarity and mitigating ambiguity issues. Additionally, QUIC

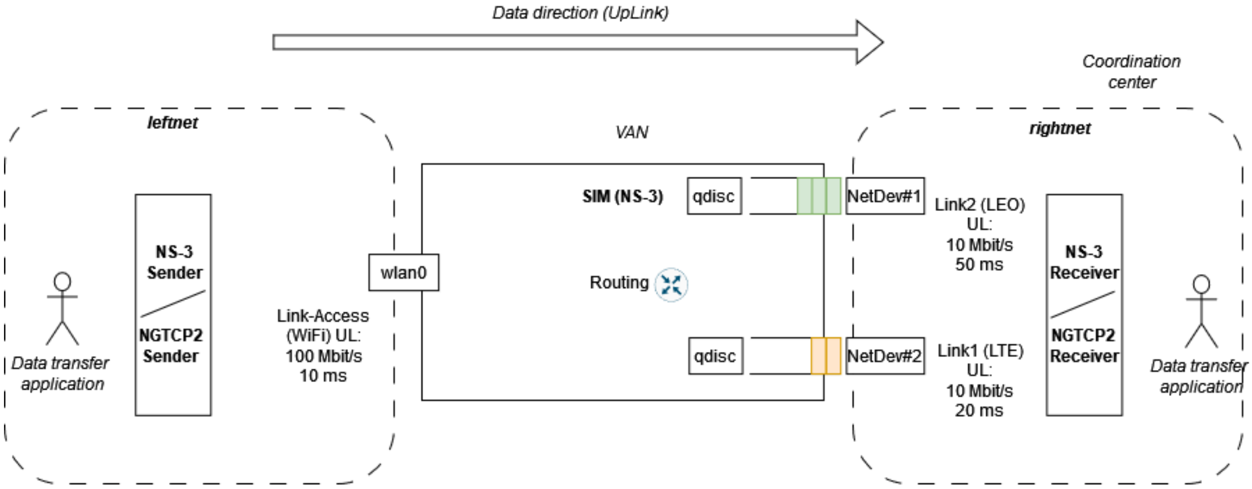


Figure 2: Emulation setup with 3 nodes and 3 links for modelling the proposed system architecture.

ACKs provide insights into the interval between packet reception and ACK generation, facilitating more accurate computation of the path Round-Trip Time (RTT). Furthermore, QUIC implements a selective acknowledgment mechanism, enabling a single QUIC ACK frame to encompass multiple ACK blocks.

Despite the partial redesign of CCA in QUIC, it remains vulnerable to macroscopic effects and potential inefficiencies of TCP NewReno CCA, particularly concerning end-to-end queue management and bottleneck characteristics [22]. These aspects underscore the ongoing evolution and refinement of QUIC's congestion control mechanisms to optimize performance in diverse network environments.

3.2. Connection migration and resilience to NAT rebinding

One of the notable benefits of utilizing UDP is its Connection (or Path) Migration feature, which distinguishes QUIC from other transport protocols. Unlike traditional network sockets, where the reference is tied to the underlying network configuration, QUIC's CID reference remains associated with the connection itself. This design choice empowers end-to-end QUIC connections and their associated streams to seamlessly adapt to path changes. In the current version of QUIC, only clients can initiate path migration, subject to security checks on the new path, as outlined in Path Validation procedures.

QUIC's consistent connection ID plays a pivotal role in facilitating uninterrupted connectivity amid changes to client IP and port configurations, such as those induced by NAT rebindings or shifts in network connectivity to a new address [23]. By maintaining a constant session key for packet encryption and decryption, QUIC automatically verifies the legitimacy of a rebound client. Consequently, the CID reference not only enables seamless migration of connections to new client IP addresses but also extends to server IP address changes, ensuring continuity across network address modifications for both client and server.

In summary, the CID reference within QUIC empowers connections to effortlessly transition to new network configurations, whether on the client or server side, bolstering the protocol's adaptability and resilience in dynamic network environments.

4. Related work

Various research work has already explored the implementation and exploitation of QUIC in various general scenarios. For instance, it has been noted that TCP generally outperforms QUIC unless there's packet loss, wherein QUIC demonstrates its advantage by mitigating the impact of head-of-line blocking [24].

In various papers the authors assess the performance of Google's variant of QUIC (gQUIC) [25, 26, 27, 28], although this variant diverges significantly from the IETF specification [29], particularly in aspects like the cryptographic handshake. Recent assessments of web performance reveal that the adoption of HTTP/3 using IETF QUIC does not consistently outperform HTTP/2, which is based on TCP [30].

Previous studies already tried to evaluate the employment of QUIC within satellite networks, although primarily focusing on Web browsing and the reduction of the Page Load Time [31, 32, 33, 34, 35]. Additionally, the longstanding recognition of Performance Enhancing Proxies (PEPs) for enhancing TCP performance over extended delay satellite links is well-documented [36]. Indeed, investigations contrasting the performance of TCP employing PEPs with QUIC suggest that it generally outperforms QUIC alone for larger data transfers [33, 34, 35].

To improve the performance of QUIC over satellite links in certain contexts, researchers have proposed some targeted adjustments, such as increasing the initial window size [34] or reducing the ACK frequency to alleviate control overhead [35]. Similarly, the QUIC BDP Frame extension was proposed to expedite throughput ramp-up during repeated connections over long-delay satellite links [37]. Finally, previous work by the authors [6] is initially addressing the QUIC protocol behavior in challenging scenarios, but only in simulation and not addressing enhanced CCAs.

In the present work, we extend the QUIC protocol performance evaluation and network optimizations in emergency scenarios in which the link in use is changing during the connections' life. Indeed, optimal application performance requires the same transport protocol to behave well also in case of significant link variations, which was not addressed in the mentioned related work. The goal of our work is then to put under stress QUIC Path Migration feature and show CCA resilience to different buffer sizes and variation of connection delays. In particular, the use of BBR as QUIC CCA is deemed important to guarantee optimal performance without over-flooding the network bottleneck, and it is an original contribution of our work with regard to the state of the art.

5. QUIC CCA performance evaluation

In this study, our objective is to evaluate different transmission and network options using QUIC protocol, for characterizing and optimizing data transfers in a realistic multi-segment communication network designed for emergency scenarios. Rather than just considering the connection's startup time and the protocol capability to reach the available channel capacity, it is also important to look at the queue occupation. In fact, a transport protocol might be very efficient in transmitting at the channel capacity, at the cost of unnecessarily flooding the queues on the end-to-end path.

A fundamental assumption influencing data transfer involves the existence of a central routing node (VAN in Figure 1), capable of supporting local wireless communications and backhaul connections through either terrestrial or satellite wireless links. Within this configuration, our focus is primarily on data uploads, assessing QUIC operational regimes under various link types. It is worth noting that while we can compare QUIC data transfer with TCP data transfer, any changes in the backhaul network cause TCP connections to reset. Depending on the application, the data transfer must be either resumed or restarted. In contrast, QUIC's path migration ensures a seamless transfer even in the event of network changes. It is important to highlight that previous work [38, 23] exploited this feature by executing a proxy operation and delivering TCP traffic within QUIC tunnels. However, in our current study, the QUIC transfer is executed end-to-end.

Concerning the queues shown in Figure 2, they are associated with the two backhaul links, representing the bottleneck links of the end-to-end path. Such queues are necessary to handle buffering at the bottleneck (according to a queuing discipline, qdisc in figure, assumed by default as First In - First Out – FIFO) before the packets are delivered to the hardware. Indeed, it is known that NewReno CCA is behaving optimally if the queue sizes are approximately equal to the Bandwidth Delay Product (BDP) [21], therefore these queues size may have different values.

It is worth to remark that smarter queuing policies other than FIFO are available, such as Controlled Delay (CoDel), PIE or Cake, [39, 22], having the goal to reduce standing buffer occupation by aggressive CCAs, mitigating the so-called *bufferbloat* effect. Therefore, as further assessment, in addition to identifying the impact of FIFO queue sizes, we will also evaluate the use of CoDel as a queuing policy [22]. Indeed, CoDel is set as the default *qdisc* in most of the recent Linux distributions.

An optimal working condition has been identified and is widely accepted by the community for CCAs, in which the ideal transmission condition is achieved when the throughput is high enough (w.r.t the bottleneck capacity) but without incurring excessive buffering in the queues [40]. In other terms, the optimal working condition is when a transmitter is able to send data close to the maximum available capacity while staying close to the minimum RTT. This key assumption is at the basis of the design of newer congestion control algorithms, namely BBR [9], but also considered to drive smart queue policies such as CoDel.

To clarify this aspect and demonstrate the possible operational positions of a CCA, we direct attention to Figure 3. This methodology draws inspiration from [41], wherein the authors present this visualization method for identifying the distinctive operational regimes of a QUIC CCA, facilitating comparisons between various implementations. In Figure 3, samples of RTT

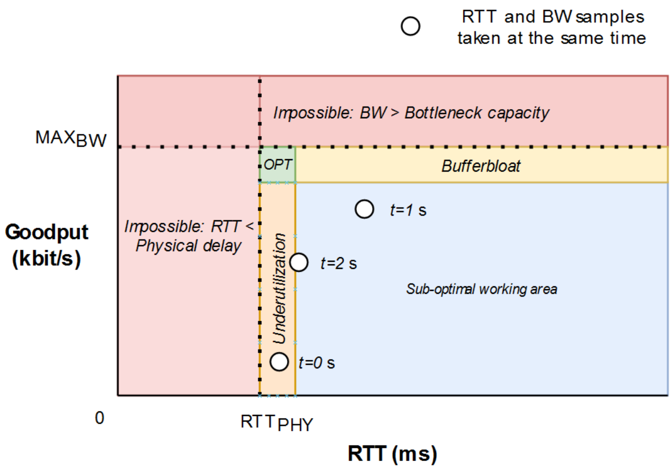


Figure 3: CCA working areas

(at the sender) and bandwidth (at the receiver) are collected at the same emulation time, and represent a dot in the figure. For clarification, three hypothetical dots are displayed representing RTT and BW samples collected at the emulation time 0 s, 1 s and 2 s. In this type of plot, the green area represents the optimal working point (*OPT*), in which the channel is fully exploited and close to its maximum available capacity, while the RTT is kept closer to the minimum one (RTT_{PHY}). This situation is optimal since the channel is well utilized, and a low RTT allows new connections to be served, especially with shorter objects to transfer, avoiding unfair conditions associated with full queues. In the red areas, we show positions where samples are not expected, since they are associated with too low RTT or too high BW, which are impossible conditions. In the *Bufferbloat* area, we expect samples of an aggressive CCA, which is leveraging high queue occupation to reach maximum capacity. In the *UnderUtilization* area instead, we expect samples with both low RTT (which is good) but also low channel utilization and consequent inefficiencies. Finally, in the blue area, we are expecting transitory samples and in general sub-optimal CCA performance.

In this first study we collect for simplicity RTT and bandwidth samples from the CCA itself (active sampling), but a more generic assessment can be done by performing measurements directly on the network and the queue occupation (passive sampling [42]), and this is left as a future work.

In the rest of the work, and in the most significant configurations, we add to the classic Goodput, CWND and RTT graphs also this new operational working area representation of the CCA under study. Of course, once the bottleneck link condition changes, such as in the case of variation of the link minimum RTT, different optimal working areas are possible.

6. Emulation environment and test campaign

To more accurately assess various QUIC CCAs across different network configurations, we opted to utilize an emulation approach employing a real open-source QUIC stack. In this way, we can produce more reliable and extensive results than what was previously assessed via simulation. The emulation environment developed for the scope is based on Docker endpoints, hosting real reference QUIC protocol stacks, with an inner Docker node to model network impairments and handover.

We relied on a customized version of the QUIC *Interop Runner* tool (QIR) Docker-based QUIC testbed found in [43]. The framework offers a test environment which can be utilized for benchmarking the performance of QUIC implementations under various network conditions. QIR makes use of the NS-3 network simulator for simulating network conditions and cross-traffic, and allows bridging the real and the simulated world. The tool allows the definition of different network scenarios, known as Quic-Network-Simulator (QNS) scenarios, and relies on a dockerized flavour of the NS-3 simulator to reproduce different network conditions, connecting different QUIC endpoints (possibly based on different implementations) between each other through the use of Docker networking facility. The network parameters depend on the specific scenario but generally include bandwidth, delay, queue size (number of packets) and loss rate.

The framework uses the docker-compose tool to build and configure three docker images: the network simulator, a QUIC client, and a QUIC server. By default, the framework uses two networks on the host machine: the *left* and *rightnet* as exemplified in Figure 2. The network named *leftnet* is connected to the client docker image, while *rightnet* is connected to the server: the NS-3 simulation sits in the middle and forwards packets between *leftnet* and *rightnet*. Left (client) node and Right (server) node will establish a QUIC connection through the link simulated by NS-3 and a file transfer (in uplink) will be performed from server to client. We have identified up-to-date parameter values for the access link data transfer and delay, allowing us to conduct a realistic evaluation of the various mechanisms. While realistic, the study is not exhaustive and does not consider recent progress in wireless connectivity (e.g., 5G) and megaconstellations (e.g., StarLink) so higher throughput channels will be tested in future work.

For the performance evaluation, we implemented a modified version of the QIR tool enhanced to contemplate additional functionality, introducing the capability to specify the queuing policy e.g., FIFO, CoDel etc., file transfer size, handover events, etc. Since there are several open-source QUIC stacks already compatible with QIR, we focused our attention on one of the stacks implementing a recent BBR CCA. At the moment of this writing, we selected the NGTCP2 QUIC stack implementation since it implements the more recent BBRv2 reference algorithms [44], while at the same time supporting standard logs (in the form of qlog [45]) allowing straightforward post-processing. For completeness, the forked version of the tool used to conduct this evaluation analysis can be found in [46].

Before performing a thorough performance evaluation of different CCA and network configurations, we also want to compare previous results from [6] with results obtained with the emulation approach. The goal is twofold: to confirm the validity of previous results and the suitability of the new emulation approach.

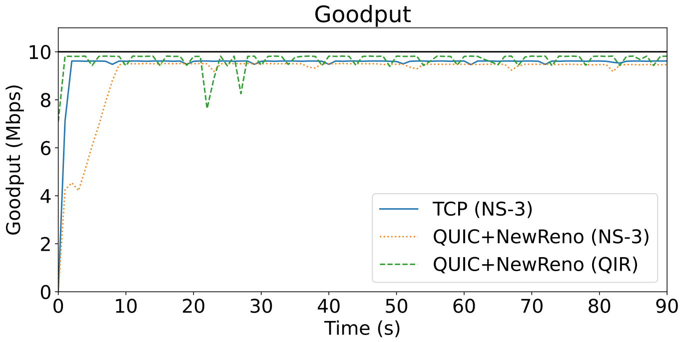
Therefore, we start by running an experimental assessment comparing the performance of the NS3-based QUIC implementation found [43], with simulated TCP and with a real end-to-end QUIC stack implementation using the modified QIR and NGTCP2 in the same network configuration (Test-1). We consider in this test a reference value for the queue size equal to the Bandwidth-Delay Product (BDP). Next, we focus on the handover behaviour considering the simulated and the real QUIC implementation, using default NewReno CCA (Test-2) with the same queue size. This assessment confirms the suitability of the emulation environment designed for its intended purpose.

After these initial validation tests, we pursue the evaluation analysis via the emulation testbed. The identified real QUIC stack (NGTCP2) includes both NewReno and BBRv2 as CCA options, which are tested in presence of network changes (i.e., an L3 handover) with different FIFO queue sizes. In individual emulation tests, we set the queue size at the central routing node respectively as i) half the BDP value of the first (terrestrial) link (Test-3) ii) the BDP value of the first (terrestrial) link (Test-4) and iii) the BDP value of the second (satellite) link (Test-5).

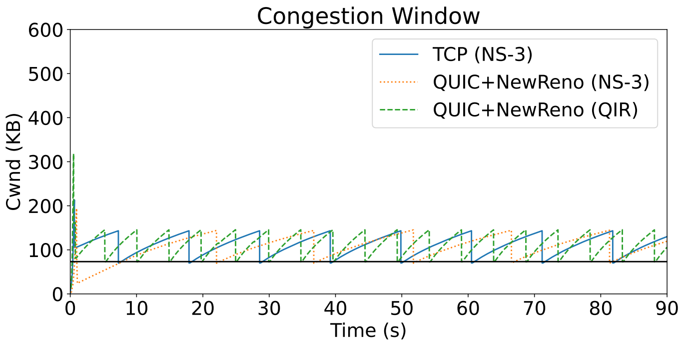
Finally, we focused on the use of CoDel smart queues, evaluating QUIC CCA NewReno (Test-6) and BBR (Test-7). In all the test configurations, the channel capacity and latency (both for the terrestrial and the satellite link) reflect the values presented in Figure 2.

7. Results

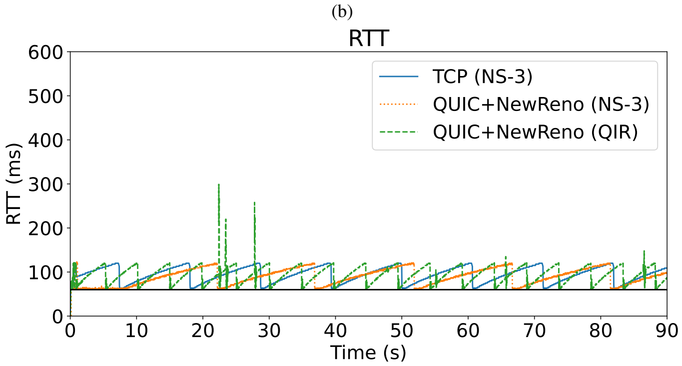
As discussed in Section 6, the first test (Test-1) compares the results obtained with the emulation framework with previous simulation results conducted with NS3 under equal network settings. In Figure 4 we can observe, starting from the throughput metric, that the overall CCA behaviour is the same, although the



(a)



(b)



(c)

Figure 4: Test-1: TCP, QUIC (NS-3) and QUIC (QIR) data uplink (a) Goodput (b) CWND (c) RTT.

emulation values are close to the TCP ones. This can be considered a better result, since it is expected for QUIC CCA to behave similarly (or better) than TCP. This small misalignment from the simulations is confirmed by the congestion window (*cwnd*) evolution graph, in which QUIC NewReno has a lower frequency of *cwnd* increase and decrease (i.e., the tooth-saw pattern). Otherwise, the overall behaviour and reaction to loss is equivalent, confirming the suitability of the proposed QIR tool.

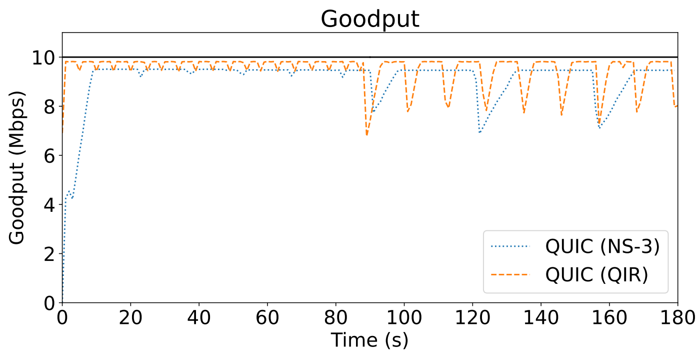
As further evaluation criteria, we decided to show the average value of RTT and its standard deviation in Table 1, with a clear alignment of all the three options under study.

To confirm the overall validity of the QIR tool, we also need to verify that both CCA versions have equivalent performance and that the tool itself is correctly implementing the required link change (handover) conditions. For this reason, in the second run (Test-2) we extended the test duration and triggered

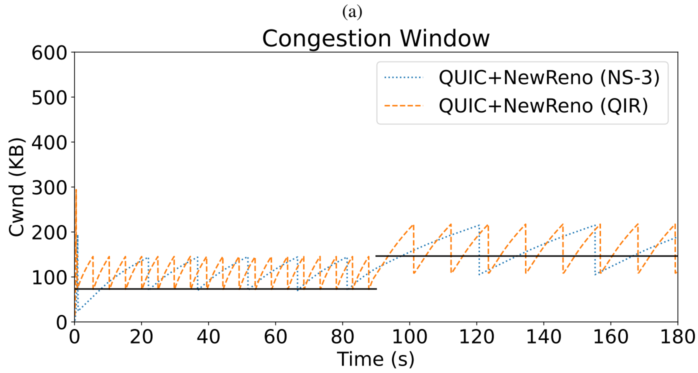
Table 1: Test-1: focus on connections RTT

Experiment	**RTT values**	
	Mean (ms)	**Std. Dev. (ms)**
TCP (NS-3)[1]	93.63	17.12
QUIC+NewReno (NS-3)[1]	89.64	18.10
QUIC+NewReno (QIR)[1]	93.46	18.13

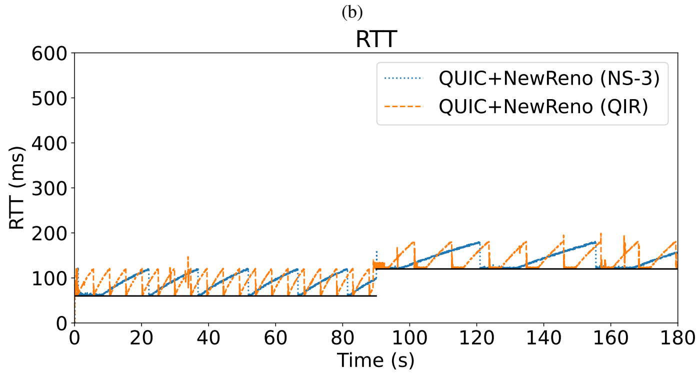
[1] The duration of the experiment is 90s. No handover. Queue size equal to the BDP of the link.



(a)



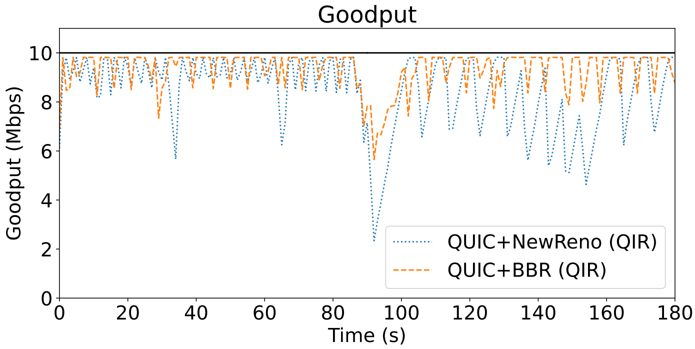
(b)



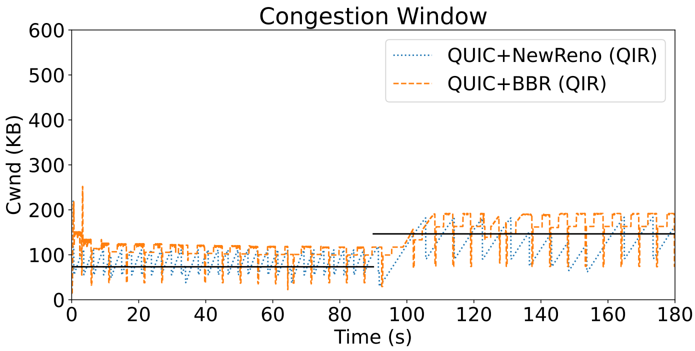
(c)

Figure 5: Test-2: QUIC+NewReno (NS-3) and QUIC+NewReno (QIR) data uplink (a) Goodput (b) CWND (c) RTT with a queue size equal to the BDP of the first link. Handover event at 90 s from the first link towards the LEO link.

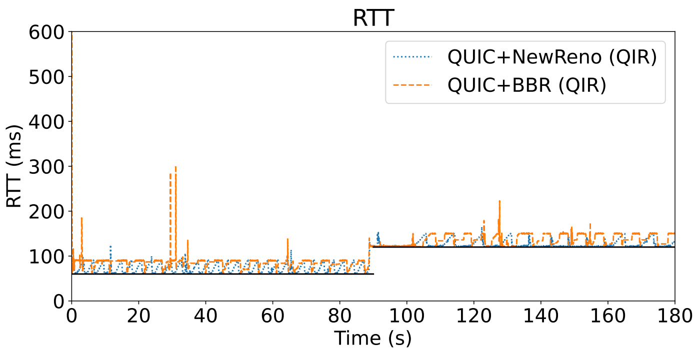
an handover event at time t (90s), from the terrestrial to the satellite link. The comparison of previous (simulation) results with the new results obtained in the emulation study is shown in Figure 5. Overall, the trend is confirmed, but again NS3 experiences a different frequency of loss events. We decided not



(a)

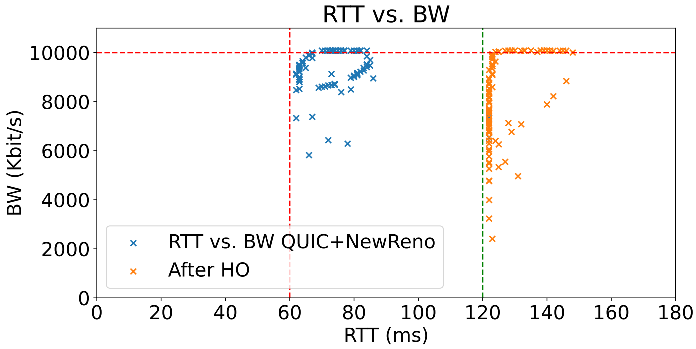


(b)

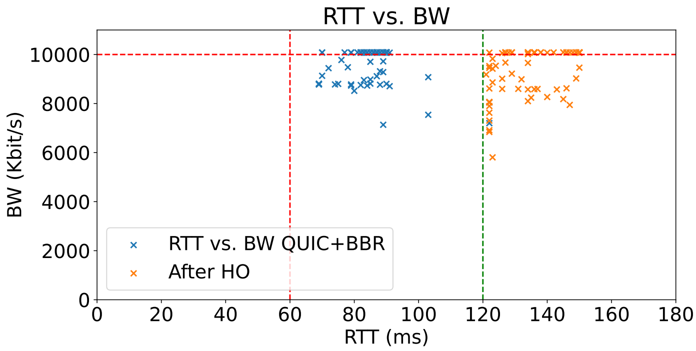


(c)

Figure 6: Test-3: QUIC+NewReno (QIR) and QUIC+BBR (QIR) data uplink (a) Goodput (b) CWND (c) RTT with a queue size equal to half of the BDP of the first link. Handover event at 90 s from the first link towards the LEO link.



(a)



(b)

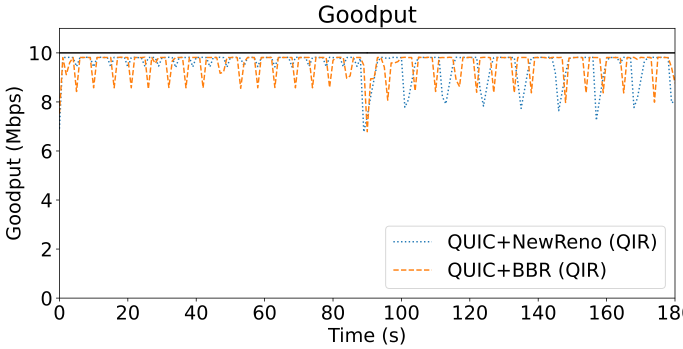
Figure 7: Test-3: RTT vs. Bandwidth

to dig any further into the NS3 QUIC source code to identify the origin of this variation, since it is an experimental project and not maintained, but we suspect that ACKs are handled differently from the standard QUIC specification. Therefore, since the overall performance is consistent (i.e., same maximum and minimum RTT, its average, the maximum and minimum *cwnd*), we can confirm that the QIR tool can be considered a reliable framework, with better performance (i.e., it works in real-time whereas the QUIC in NS3 resulted slower) and it allows testing real-life CCAs with more realistic setups.

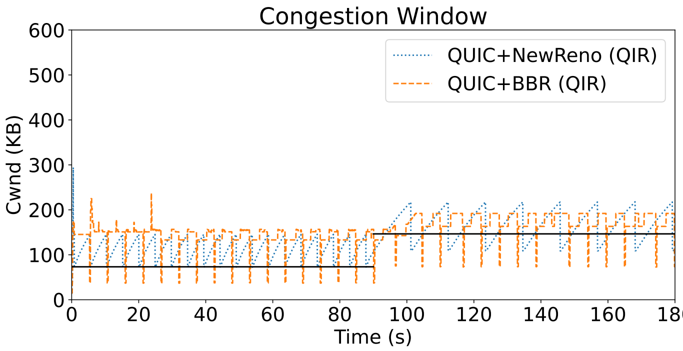
In the following tests, we conduct an emulation-based comparison of different CCAs starting with NewReno CCA (default CCA as defined in [14]) and then BBRv2. As BBRv2 is specifically tailored for shallow buffers, it is valuable to evaluate its performance in the target scenario and compare it with NewReno. To this end, we start by measuring the classic throughput–*cwnd*–RTT graphs as a function of time for Test-3, included in Figure 6. While the overall RTT is quite similar, we can notice a marginal improvement in channel utilization by BBR.

Since we observed some differences between NewReno and BBRv2 CCA dynamics, we zoom in by analysing the working areas plots. In Figure 7, are shown the working BW/RTT value pairs for NewReno and BBRas CCA. In these plots, we have two working reference areas' highlighted by the red and the green reference lines. Due to the handover, there is no bandwidth change (set to 10 Mbit/s) but just an increase of the minimum RTT (two-way link latency) from 60 to 120 ms. In Figure 7a we see the results for NewReno. Due to the relatively small buffer, it is expected for the protocol to have several samples below the maximum channel capacity. This is particularly evident after the handover (orange crosses), since the new link is characterized by an even higher BDP value. In summary, by using NewReno we have mainly a combination of Bufferbloat and Underutilization working areas. Instead, when using BBRv2 as shown in Figure 7b, BW samples in general have higher values and with just marginal increases of RTT. This behaviour indeed represents an advantage in transmission and results in a more effective data transfer when compared to NewReno.

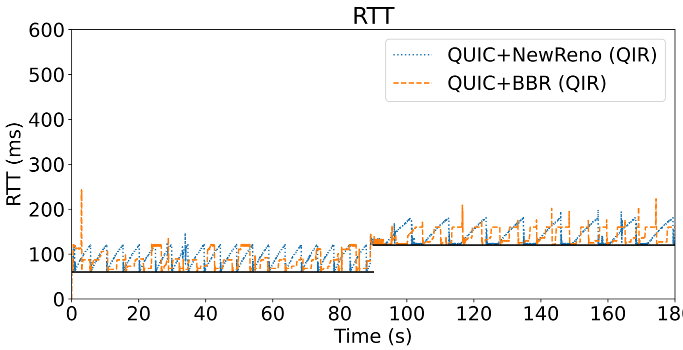
Going ahead with the planned tests, we can focus on CCA comparison when the bottleneck has a queue size equal to the BDP of the first link (Test-4). In this case, shown in Figure 8, both CCAs show good behaviour in terms of bandwidth, with



(a)

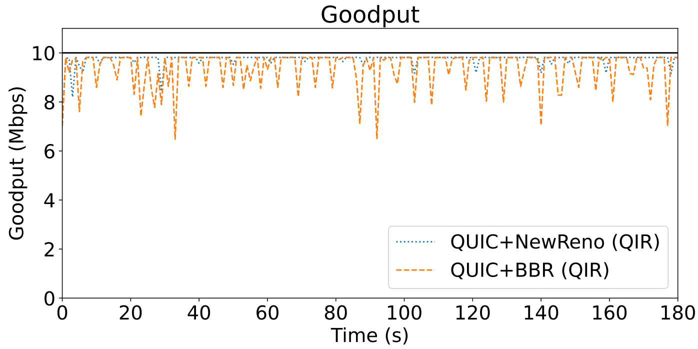


(b)

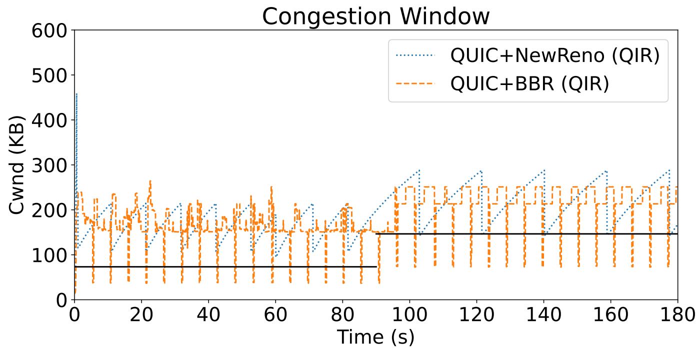


(c)

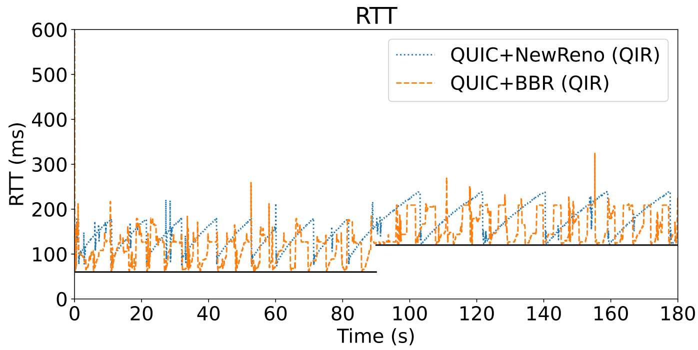
Figure 8: Test-4: QUIC+NewReno (QIR) and QUIC+BBR (QIR) data uplink (a) Goodput (b) CWND (c) RTT with a queue size equal to the BDP of the first link. Handover event at 90 s from the first link towards the LEO link.



(a)



(b)



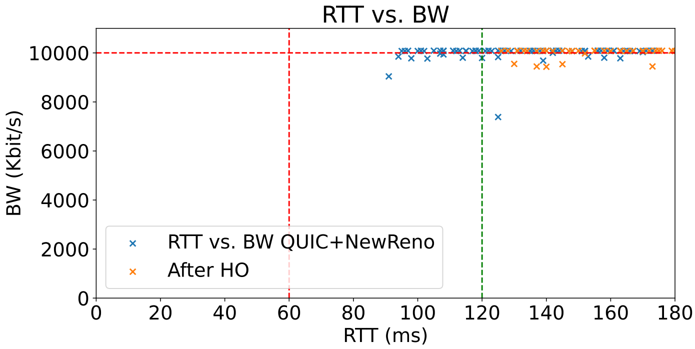
(c)

Figure 9: Test-5: QUIC+NewReno (QIR) and QUIC+BBR (QIR) data uplink (a) Goodput (b) CWND (c) RTT with a queue size equal to the BDP of the second link. Handover event at 90 s from the first link towards the LEO link.

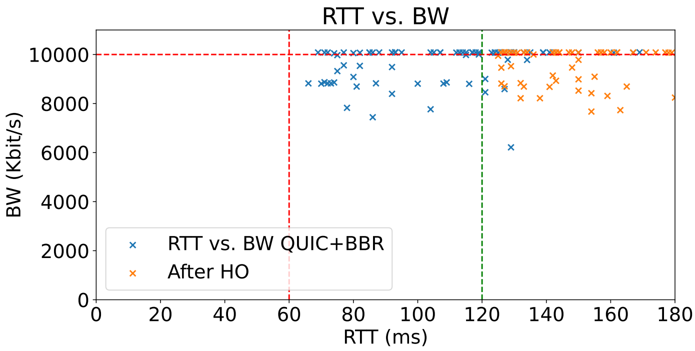
BBR exhibiting a lower RTT and consequent lighter impact on the queues. For this setting, we omit the working area plot for the sake of brevity, but it is clear that we have a marginal improvement in network utilization and fairness in resource occupation by BBR, both before and after the handover. A similar view is presented later on when we conduct the comparison with CoDel (Test-6 and Test-7). Indeed, the BBRv2 response is very similar to the half-BDP case, therefore confirming its suitability in wider ranges of network configurations.

In the last test dealing with different FIFO queue sizes (Test-5), we consider for its value the BDP of the slowest link (i.e., an higher value than the BDP of the fastest link). In such conditions, a more evident improvement of BBR when compared to NewReno is experienced. This behaviour is to be attributed to a more accurate bandwidth estimation by BBR which also avoids a too high increase of the *cwnd*. From an analysis of throughput–cwnd–RTT plots in Figure 9 and in comparison with the previous tests, we confirm that BBR overall dynamics is not dependent on the queue size, as the protocol is self-controlling its transmission rate without overfilling the queues. Instead, NewReno introduced a more severe pressure on the queues with an artificial increase of the network latency (bufferbloat), which is detrimental for competing flows sharing the same link.

For BBR, while it is very effective in containing the RTT before the handover event, after the handover event, it suffers from some adaptation problems which shall be further investigated. This observation is confirmed by looking at the working areas plot of Figure 10, showing an activity closer to the optimal areas for BBR before the handover, whereas after the handover both



(a)



(b)

Figure 10: Test-5: RTT vs. Bandwidth – (a) NewReno and (b) BBR

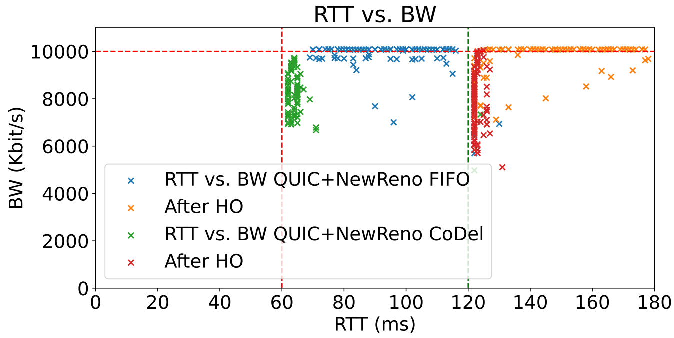


Figure 11: Test-6: RTT vs. Bandwidth – NewReno FIFO vs CoDel

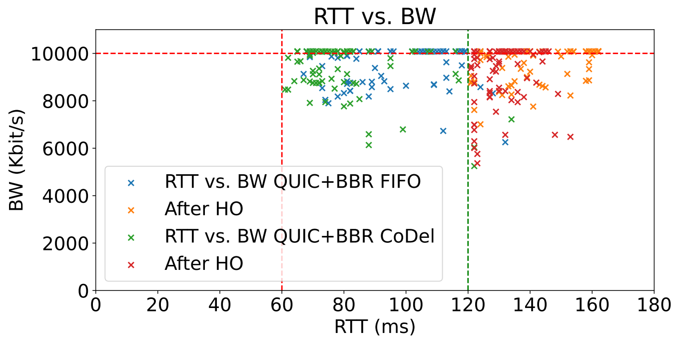


Figure 12: Test-7: RTT vs. Bandwidth – NewReno FIFO vs CoDel

CCAs under investigation are equally putting excessive pressure on the queues.

The conclusion is that, contrary to what we saw before, BBR is not a clear winner in terms of performance in critical networking conditions, even though it is able to reduce bufferbloat at the cost of a slightly lower bandwidth occupation (see for instance the blue crosses in Figure 10). For this reason, we shall continue the protocol characterization and performance evaluation also in presence of different networking conditions, and in particular the use of smart queues (CoDel).

In Test-6 we compare NewReno CCA when using a FIFO queue with a size equal to the BDP of the first link (i.e., in the same configuration as the one in Test-4) versus the use of a CoDel queue instead. By an analysis of the results in Figure 11, we can observe that CoDel is able to reduce CCA BW-RTT value pairs in the bufferbloat area at the cost of having such samples now positioned into the underutilization working areas (i.e., blue and orange samples become respectively the green and red ones). In other terms, while with FIFO the NewReno CCA is continuously pushing on the buffers resulting in an overall RTT increase, CoDel is preventing this condition by triggering targeted losses on the communication as soon as a too high standing queue is detected. Of course, the transmission in average will happen when CoDel is used at a lower bitrate.

On the contrary for BBRv2 CCA, in case of Test-7 (see Figure 12), we observe an overall overlap of BW-RTT samples regardless of the queue management policy. This is to be expected since BBR is working on similar principles to CoDel, preventing losses by an accurate estimation of the sender rate. In this respect, when considering the outcomes of these latter two tests, the use of BBR is to be preferred also in presence of smart queues.

Nonetheless, in general, the use of CoDel is suitable to control the delay and make the network interactive and responsive (owing to the low RTT), partially sacrificing the goodput, and it represents a single queue configuration for both links and for both CCAs under study.

In conclusion, the presented test campaign allowed us to identify a suitable QUIC protocol and queue configurations in challenging and variable network conditions. The analysis is based on commodity link configuration options (i.e., fixed speed and latency with only two main configurations for terrestrial and LEO), handover events (single handover), full availability of the links (no significant/random loss events or sudden and temporary link outages), and considering simple application models (uplink only). Furthermore, smart queue configurations can be explored with a wider set of possible values, paired with a broader coverage of CCA options available, in future work.

Nonetheless, the results presented herein already provide very useful hints and characterization of QUIC protocol operations in mixed terrestrial-satellite networks. Even though BBR can be considered the actual state of the art, and it appeared as the most flexible solution so far, it is currently being upgraded in its recent BBRv3 version, which shall be carefully analysed as well.

Furthermore, alternative CCAs for QUIC specifically tailored for communication in harsh/emergency environments can be defined, also based on alternative approaches such TCP

Table 2: From Test-2 to Test-7: focus on Goodput

Experiment	Ref. Fig.	First Link		Second Link	
		Mean (Mbps)	Std. Dev. (Mbps)	Mean (Mbps)	Std.Dev. (Mbps)
QUIC+NewReno (NS-3)[1]	5	9.06	1.46	9.05	0.71
QUIC+NewReno (QIR)[1]	5, 8	9.65	0.51	9.35	0.72
QUIC+BBR (QIR)[1]	8	9.51	0.58	9.54	0.54
QUIC+NewReno (QIR)[2]	6, 7	8.89	1.20	7.53	1.72
QUIC+BBR (QIR)[2]	6, 7	9.22	0.92	8.98	1.24
QUIC+NewReno (QIR)[3]	9, 10	9.72	0.36	9.77	0.15
QUIC+BBR (QIR)[3]	9, 10	9.34	0.76	9.43	0.73
QUIC+NewReno (QIR)[4]	11	7.84	1.03	7.57	1.36
QUIC+BBR (QIR)[4]	12	9.19	1.05	8.92	1.24

[1] The duration of the experiment is 180s. Handover event at 90s from the first link to the second link. Queue size equal to the BDP of the first link.
[2] The duration of the experiment is 180s. Handover event at 90s from the first link to the second link. Queue size equal to half of the BDP of the first link.
[3] The duration of the experiment is 180s. Handover event at 90s from the first link to the second link. Queue size equal to the BDP of the second link.
[4] The duration of the experiment is 180s. Handover event at 90s from the first link to the second link. Adaptive CoDel queue (interval 100ms, target 5ms).

Table 3: From Test-2 to Test-7: focus on RTT

Experiment	Ref. Fig.	First Link		Second Link	
		Mean (ms)	Std. Dev. (ms)	Mean. Dev. (ms)	Std. Dev. (ms)
QUIC+NewReno (NS-3)[1]	5	89.64	18.10	141.28	18.66
QUIC+NewReno (QIR)[1]	5, 8	93.42	17.25	144.64	19.62
QUIC+BBR (QIR)[1]	8	85.04	18.72	141.46	16.16
QUIC+NewReno (QIR)[2]	6, 7	73.26	13.19	126.80	9.86
QUIC+BBR (QIR)[2]	6, 7	76.77	13.63	138.97	16.22
QUIC+NewReno (QIR)[3]	9, 10	135.92	26.91	186.93	33.71
QUIC+BBR (QIR)[3]	9, 10	107.24	36.41	145.50	20.47
QUIC+NewReno (QIR)[4]	11	65.40	11.76	123.40	34.46
QUIC+BBR (QIR)[4]	12	77.83	16.67	131.88	9.83

[1] The duration of the experiment is 180s. Handover event at 90s from the first link to the second link. Queue size equal to the BDP of the first link.
[2] The duration of the experiment is 180s. Handover event at 90s from the first link to the second link. Queue size equal to half of the BDP of the first link.
[3] The duration of the experiment is 180s. Handover event at 90s from the first link to the second link. Queue size equal to the BDP of the second link.
[4] The duration of the experiment is 180s. Handover event at 90s from the first link to the second link. Adaptive CoDel queue (interval 100ms, target 5ms).

Wave [47, 48]. In particular, the design of a satellite-tailored CCA, which is robust to link changes and avoids pressure on the queues, is being defined within the scope of the QUICoS project [49].

As a final recap of all the tests performed, Table 2 provides a simple and uniform view of all the goodput performance indicators for all the handover scenarios (Test-3 to Test-7), and in Table 3 the corresponding measured RTT values. For the sake of clarity, for each line in the table we also report (in column *Ref. Fig.*) the reference to the figure in the paper where the outcome of that test is shown.

8. Conclusion

In this work, we introduce and study the possible use of a dual-stack node architecture along with the implementation of the QUIC transport protocol for emergency communications, assuming that the backhaul link used to transfer users' applications data needs to be changed seamlessly. After an initial comparison with previous stimulative tools for QUIC and TCP, the work deals with an extensive performance evaluation using an emulation environment using real open-source QUIC stacks and Docker containers. The use of an emulation approach allowed us to experience a wider set of realistic protocol and network configurations, with the aim of identifying operational QUIC's CCAs characteristics and dynamics. In this setting, QUIC demonstrated high reliability and efficiency, even if in some cases it showed some underperforming conditions. At the same time, CoDel smart queues can help to lower latency communications at the cost of reducing the transmission bitrates. Optimal working conditions of maximum Bandwidth exploitation and minimum RTT increase are difficult to achieve, and both BBR and CoDel were designed with this goal in mind.

In conclusion, considering all the evaluations, no solution is a silver bullet for effective data transmission in challenging network conditions. Nonetheless, BBR proved to be more effective in most of the conditions, without suffering impairments when CoDel is employed.

From these results, we shall consider in future work additional CCAs and queue configurations, including severe loss models, higher capacity and more asymmetry in the communication. Eventually, a customized CCA for QUIC applicable to the reference scenario will be considered and assessed, as an outcome of the acknowledged QUICoS project. Furthermore, beyond the efficacy, we also plan to evaluate the fairness and friendliness of the proposed solution in case of heterogeneous competing flows.

Finally, as mentioned, we utilized an emulator to analyze the performance of the scenario and technology under consideration. Emulators occupy a middle ground between real experiments and simulations. They offer greater realism than simulators while still allowing for full control over experimental configurations (bandwidth, latency, errors, concurrent traffic, etc.). However, emulators may not fully replicate the diverse hardware characteristics, unpredictable challenges, and user behaviors encountered in real-world networks. Therefore, while the performance outcomes and trends observed with emulators are generally reliable, we plan to conduct real-world experiments prior to suggest actual deployment of our solution.

Acknowledgments

The scenarios and results presented in this work are part of the outcomes of the European Space Agency's (ESA) project "QUIC over Satellite" (QUICoS [49]) contract n. 4000138640/22/NL/AF. Responsibility of the contents resides with the authors.

References

[1] P. K. Esubonteng, R. Rojas-Cessa, RESTORE: Low-Energy Drone-Assisted NLoS-FSO Emergency Communications, IEEE Access 10 (2022).

[2] S. D'Auria, M. Luglio, C. Roseti, R. Strollo, F. Zampognaro, Real time Transmission of Cultural Heritage 3D Survey in Case of Emergency, in: Proc. of the International Conference on Information and Communication Technologies for Disaster Management, 2016.

[3] F. Pervez, J. Qadir, M. Khalil, T. Yaqoob, U. Ashraf, S. Younis, Wireless Technologies for Emergency Response: A Comprehensive Review and Some Guidelines, IEEE Access 6 (2018).

[4] M. C. Filippou, D. Sabella, M. Emara, S. Prabhakaran, Y. Shi, B. Bian, A. Rao, Multi-Access Edge Computing: A Comparative Analysis of 5G System Deployments and Service Consumption Locality Variants, IEEE Communications Standards Magazine 4 (2) (2020) 32–39.

[5] M. Luglio, M. Marchese, F. Patrone, C. Roseti, F. Zampognaro, Performance Evaluation of a Satellite Communication-Based MEC Architecture for IoT Applications, IEEE Transactions on Aerospace and Electronic Systems 58 (5) (2022) 3775 – 3785.

[6] A. Bujari, M. Franco, C. E. Palazzi, M. Quadrini, C. Roseti, F. Zampognaro, Use of QUIC Protocol for Efficient Data Transmission Over Satellite in Emergency Scenario, in: Proc. of the International Conference on Information and Communication Technologies for Disaster Management, 2023, pp. 1–6.

[7] J. Iyengar, M. Thomson, QUIC: A UDP-Based Multiplexed and Secure Transport, RFC 9000 (May 2021).
URL https://www.rfc-editor.org/info/rfc9000

[8] M. Bishop, HTTP/3, RFC 9114 (Jun. 2022). doi:10.17487/RFC9114.
URL https://www.rfc-editor.org/info/rfc9114

[9] N. Cardwell, Y. Cheng, C. S. Gunn, S. H. Yeganeh, V. Jacobson, BBR: Congestion-Based Congestion Control, ACM Queue 14, September-October (2016) 20 – 53.
URL http://queue.acm.org/detail.cfm?id=3022184

[10] Q. D. Coninck, O. Bonaventure, Multipath Extensions for QUIC (MP-QUIC), Internet-Draft draft-deconinck-quic-multipath-07, Internet Engineering Task Force, work in Progress (2021).
URL https://datatracker.ietf.org/doc/draft-deconinck-quic-multipath/07/

[11] M. Albulet, Spacex Non-geostationary Satellite System: Attachment a Technical Information to Supplement Schedules, US Fed. Commun. Comm. (2016).

[12] F. De Rango, N. Palmieri, A. F. Santamaria, G. Potrino, A Simulator for UAVs Management in Agriculture Domain, in: Proc. of SPECTS, 2017, p. 1–8.

[13] C. Prandi, S. Mirri, S. Ferretti, P. Salomoni, On the Need of Trustworthy Sensing and Crowdsourcing for Urban Accessibility in Smart City , ACM Transactions on Internet Technology 18 (1) (2021) 1–21.

[14] J. Iyengar, I. Swett, QUIC Loss Detection and Congestion Control, RFC 9002 (May 2021).
URL https://www.rfc-editor.org/info/rfc9002

[15] A. Gurtov, T. Henderson, S. Floyd, Y. Nishida, The NewReno Modification to TCP's Fast Recovery Algorithm, RFC 6582 (Apr. 2012). doi:10.17487/RFC6582.
URL https://www.rfc-editor.org/info/rfc6582

[16] A. L. Grieco, S. Mascolo, Performance Evaluation and Comparison of Westwood+, New Reno, and Vegas TCP Congestion Control, SIGCOMM Computer Communications Review 34 (2) (2004) 25–38.

[17] C. Caini, R. Firrincieli, TCP Hybla: a TCP Enhancement for Heterogeneous Networks, International Journal of Satellite Communications and Networking 22 (5) (2004) 547–566.

[18] G. Marfia, C. E. Palazzi, G. Pau, M. Gerla, M. Roccetti, TCP Libra: Derivation, Analysis, and Comparison with other RTT-fair TCPs, Computer Networks 54 (14) (2010) 2327–2344.

[19] L. S. Brakmo, S. W. O'Malley, L. L. Peterson, TCP Vegas: New Techniques for Congestion Detection and Avoidance, in: SIGCOMM, 1994.

[20] S. Ha, I. Rhee, L. Xu, CUBIC: A New TCP-Friendly High-Speed TCP Variant, SIGOPS Operating System Review 42 (5) (2008) 64–74.

[21] W. R. Stevens, TCP/IP Illustrated (Vol. 1): The Protocols, Addison-Wesley Longman Publishing Co., Inc., USA, 1993.

[22] F. Zampognaro, Enabling CoDel AQM with TCP Cubic connections over Satellite Links, in: Proc. of the International Symposium on Advanced Electrical and Communication Technologies, 2019, pp. 1–6.

[23] M. Luglio, M. Quadrini, S. P. Romano, C. Roseti, F. Zampognaro, Enabling an Efficient Satellite-Terrestrial Hybrid Transport Service through a QUIC-based Proxy Function, in: Proc. of the International Symposium on Networks, Computers and Communications, 2020, pp. 1–6.

[24] D. Saif, C.-H. Lung, A. Matrawy, An Early Benchmark of Quality of Experience Between HTTP/2 and HTTP/3 Using Lighthouse, in: ICC 2021 - IEEE International Conference on Communications, 2021, pp. 1–6.

[25] A. M. Kakhki, S. Jero, D. Choffnes, C. Nita-Rotaru, A. Mislove, Taking a Long Look at QUIC: An Approach for Rigorous Evaluation of Rapidly

Evolving Transport Protocols, in: Proc. of Internet Measurement Conference, IMC '17, Association for Computing Machinery, New York, NY, USA, 2017, p. 290–303.
[26] G. Carlucci, L. De Cicco, S. Mascolo, HTTP over UDP: An Experimental Investigation of QUIC, in: Proc. of ACM Symposium on Applied Computing, SAC '15, Association for Computing Machinery, New York, NY, USA, 2015, p. 609–614.
[27] Y. Yu, M. Xu, Y. Yang, When QUIC Meets TCP: An Experimental Study, in: 2017 IEEE 36th International Performance Computing and Communications Conference (IPCCC), 2017, pp. 1–8.
[28] S. Cook, B. Mathieu, P. Truong, I. Hamchaoui, QUIC: Better for What and for Whom?, in: 2017 IEEE International Conference on Communications (ICC), 2017, pp. 1–6.
[29] M. Nottingham, What's Happening with QUIC, `https://www.ietf.org/blog/whats-happening-quic/` (2018).
[30] A. Yu, T. A. Benson, Dissecting Performance of Production QUIC, in: Proc. of Web Conference, WWW '21, New York, NY, USA, 2021, p. 1157–1168.
[31] S. Yang, H. Li, Q. Wu, Performance Analysis of QUIC Protocol in Integrated Satellites and Terrestrial Networks, in: 2018 14th International Wireless Communications & Mobile Computing Conference (IWCMC), 2018, pp. 1425–1430.
[32] Zhang, Han *et al*, How Quick Is QUIC in Satellite Networks, in: Communications, Signal Processing, and Systems, Springer Singapore, Singapore, 2019, pp. 387–394.
[33] L. Thomas, E. Dubois, N. Kuhn, E. Lochin, Google QUIC Performance over a Public SATCOM Access, International Journal of Satellite Communications and Networking 37 (6) (2019) 601–611. arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1002/sat.1301.
[34] N. Kuhn, F. Michel, L. Thomas, E. Dubois, E. Lochin, QUIC: Opportunities and Threats in SATCOM, in: 2020 10th Advanced Satellite Multimedia Systems Conference and the 16th Signal Processing for Space Communications Workshop (ASMS/SPSC), 2020, pp. 1–7.
[35] A. Custura, T. Jones, G. Fairhurst, Impact of Acknowledgements Using IETF QUIC on Satellite Performance, in: 2020 10th Advanced Satellite Multimedia Systems Conference and the 16th Signal Processing for Space Communications Workshop (ASMS/SPSC), 2020, pp. 1–8.
[36] C. Caini, P. Cornice, R. Firrincieli, M. Livini, D. Lacamera, TCP, PEP and DTN Performance on Disruptive Satellite Channels, in: 2009 International Workshop on Satellite and Space Communications, 2009, pp. 371–375.
[37] N. Kuhn, F. Simo, D. Pradas, E. Stephan, Evaluating BDP FRAME Extension for QUIC (2021). arXiv:2112.05450.
[38] A. Abdelsalam, M. Luglio, M. Quadrini, C. Roseti, F. Zampognaro, QUIC-Proxy Based Architecture for Satellite Communication to Enhance a 5G Scenario, in: Proc. of the International Symposium on Networks, Computers and Communications, 2019.
[39] G. Carlucci, L. De Cicco, S. Mascolo, Controlling Queuing Delays for Real-Time Communication: The Interplay of E2E and AQM Algorithms, SIGCOMM Comput. Commun. Rev. 46 (3) (jul 2018).
[40] L. Kleinrock, Internet Congestion Control Using the Power Metric: Keep the Pipe just Full, but no Fuller, Ad Hoc Networks 80 (2018) 142–157.
[41] A. Mishra, S. Lim, B. Leong, Understanding Speciation in QUIC Congestion Control, in: Proc. of the ACM Internet Measurement Conference, IMC '22, Association for Computing Machinery, New York, NY, USA, 2022, p. 560–566.
[42] A. Bujari, A. Marin, C. E. Palazzi, S. Rossi, Analysis of ECN/RED and SAP-LAW with Simultaneous TCP and UDP Traffic, Computer Networks 108 (2016) 160–170.
[43] `https://github.com/quic-interop/quic-network-simulator`.
[44] N. Cardwell, Y. Cheng, C. S. Gunn, S. H. Yeganeh, I. Swett, V. V., BBRv2: A Model-based Congestion Control, in: Proc. of the IETF 102th Meeting, 2018.
[45] R. Marx, L. Niccolini, M. Seemann, L. Pardue, Main Logging Schema for qlog, Internet-Draft draft-ietf-quic-qlog-main-schema-08, Internet Engineering Task Force, work in Progress (2024).
URL `https://datatracker.ietf.org/doc/draft-ietf-quic-qlog-main-schema/08/`
[46] `https://gitlab.com/romars1/quicos/quic_devel/`.
[47] A. Abdelsalam, M. Luglio, N. Patriciello, C. Roseti, F. Zampognaro, TCP Wave over Linux: A Disruptive Alternative to the Traditional TCP Window Approach, Computer Networks 184 (2021) 107633.
[48] A. Abdelsalam., M. Luglio., C. Roseti., F. Zampognaro., TCP Wave Resilience to Link Changes - A New Transport Layer Approach Towards Dynamic Communication Environments, in: Proc. of International Joint Conference on e-Business and Telecommunications, INSTICC, SciTePress, 2016, pp. 72–79.
[49] ESA Artes, End-To-End Congestion Control For Broadband Networks With Satellite-Tailored QUIC Protocol, `https://connectivity.esa.int/projects/quicos` (2024).