Careful Resume: Design and Analysis with Picoquic over Satellite Paths

Matthias Hofstätter[1], Jörg Deutschmann[1], Raffaello Secchi[2], Gorry Fairhurst[2], Reinhard German[1]
[1]*Computer Networks and Communication Systems, Friedrich-Alexander-Universität Erlangen-Nürnberg, Germany*
matthias.hofstaetter@fau.de, joerg.deutschmann@fau.de, reinhard.german@fau.de
[2]*Department of Engineering, University of Aberdeen, UK*
r.secchi@abdn.ac.uk, g.fairhurst@abdn.ac.uk

***Abstract*—Slow Start increases the congestion window by doubling each round trip until congestion is detected. As a result, reaching optimal speed typically takes several round trip times, especially important in high bandwidth-delay product networks. Geostationary satellite communications now use Performance Enhancing Proxies to accelerate the growth of the congestion window. However, these are ineffective for connections with encrypted headers or UDP traffic. We propose a new method, Careful Resume, that enables a rapid ramp-up of congestion control by reusing previous parameters. We implement Careful Resume in Picoquic and run performance tests using an emulated geostationary satellite path, three real geostationary satellite networks, the low Earth orbit satellite megaconstellation Starlink, and a cellular 5G network. Results confirm that Careful Resume accelerates the ramp-up process, reducing total transmission time. Even for connections with lower bandwidth-delay products, the network congestion point can be reached much more quickly. Benefits are particularly pronounced for paths with high bandwidth-delay products and medium-sized transfers such as web traffic.**

***Index Terms*—Careful Resume, Congestion Control, Transport Protocol, QUIC, Slow Start.**

I. Introduction

The recently standardised IETF QUIC [1] is maturing as it becomes widely deployed across the Internet in web servers and clients. Compared to the traditional TCP/HTTP stack, QUIC offers new features that improve the performance of web applications and streaming services [2]. These include support for multiple bidirectional streams, faster connection setup, and flexible reconfiguration. In addition, QUIC incorporates TLS encryption at the transport layer to protect integrity and confidentiality [3].

The systematic use of encryption in QUIC has introduced new transport-related challenges in satellite networking [4]. An Internet path using a satellite system can encounter characteristics that differ from terrestrial paths (including additional path delay, and radio impairments). One common way to mitigate these characteristics is to employ Performance Enhancing Proxies (PEPs) [5] in geostationary orbit (GEO) satellite networks. PEPs terminate a transport connection at the terminal and gateway, respectively, and apply an optimised transport method to the satellite segment. This requires access to the transport header information, e.g., the TCP packet header. Specifications to enable PEPs to interact with QUIC are currently being developed [6]. However, these require configuration of intermediate nodes and access to the end-to-end shared secrets that may not always be feasible or desirable.

Without PEPs, QUIC transport depends solely on the end-to-end protocol mechanisms. Algorithms such as congestion control (CC) are not optimised for high-latency environments. Performance limitations are particularly felt at startup when the permitted rate is controlled by Slow Start, or one of its variants, as it grows the rate to the path capacity. Since this requires multiple round trip times (RTTs) to achieve efficient capacity utilisation, poor performance has been observed with QUIC over high bandwidth-delay product (BDP) paths [7]–[11].

This paper describes how Careful Resume [12], a modification of the standard Internet slow start, can significantly reduce the number of RTTs required for startup. The idea is to use available capacity estimates from previous connections to efficiently restart transmission on the same path, essentially skipping the lengthy Slow Start process.

While the idea of reusing path information from previous connections is not new [13], [14], our approach introduces enhancements that stabilise the rate in response to a potential change in path characteristics or traffic conditions. First, we validate the path RTT by comparing RTT samples from the current connection to previously recorded RTT values, preventing aggressive rate increases when inconsistencies are detected. Additionally, we introduce a *Safe Retreat* phase to rapidly reduce the rate to an appropriate level if congestion is detected following a rate acceleration. During Safe Retreat, we use ACK information to estimate available capacity and quickly converge at a safe congestion window (CWND) after recovery.

[15] sets goals for CC evaluation, but does not specify specific evaluation methods. It encourages multiple approaches to evaluation and the use of Internet-scale measurement and acknowledges that the most appropriate approach depends on the criteria being evaluated and the maturity of the specification.

In this paper, we evaluate the performance of Careful Resume across various networks, including satellite networks (GEO and LEO) and terrestrial Internet access via cellular 5G connections. Our results show that compared to Slow Start and HyStart [16], the most widespread startup behaviour on the Internet, our approach provides benefits in a reduced completion time with minimal disruption for concurrent connections across a range of transfer sizes. Compared to methods that restart at a previously reached high rate without RTT validation or Safe Retreat, Careful Resume avoids unwanted effects when there is a significant change in path characteristics.

This paper also shows that a Careful Resume flow can coexist with other connections without significant performance degradation or starving the sharing flows, closely resembling the profile of the Slow Start phase. Conversely, it is shown that relying solely on an increased initial congestion window (IW) results in persistent congestion and losses when there are significant changes in path characteristics.

Initially, the paper evaluates the behaviour of a single CC algorithm to evaluate one or more flows that share a bottleneck link (i.e., with no flows using a differing CC algorithm). The remainder of the paper is structured as follows: Section II describes related work, and Section III outlines the design of Careful Resume. Using the implementation from Section IV and the testbed from Section V, results are shown in Section VI. Section VII concludes this paper and gives directions for future work.

II. Related Work

CC algorithms [15] seek a balance between allowing a single flow to use as much capacity as possible and limiting the sending rate to avoid congestion collapse, while also seeking to share the network capacity between competing flows. This is typically achieved by maintaining a CWND, which is the upper limit of the number of bytes in flight. The CWND is modulated based on received acknowledgements (ACKs) and other signals indicative of congestion.

The CWND starts constrained by a small IW and is increased while the sender thinks it is safe to send more. Standard TCP CC [17] uses Slow Start to determine path capacity at startup. This involves doubling the transmission rate with each RTT until congestion is detected (e.g., packet loss or reception of a Congestion-Experienced signal). The Slow Start threshold (ssthresh) is used to record the CWND at which the congestion is detected. A drawback is that this can take several RTTs for a high-capacity path, leading to underutilisation of capacity while the CWND remains below the BDP. Furthermore, the exponential increase of the CWND often leads to a significant overshoot of the path capacity, causing substantial packet loss and triggering further congestion episodes.

Several mechanisms have been proposed to avoid overshoot by dynamically adjusting the transmission rate at startup. Limited Slow Start [18] has a more cautious ramp-up of the CWND in response to initial feedback. HyStart [16] and HyStart++ [19] enhance Slow Start by monitoring RTT samples and adaptively reducing the ssthresh in response to increases in latency. This proactive adjustment helps prevent excessive CWND growth. Bottleneck Bandwidth Rate (BBR) [20] is an alternative CC algorithm that optimises CWND by estimating the bottleneck bandwidth and RTT for more rapid ramp-up without overshooting. While these CC algorithms are effective for terrestrial networks, recent research indicates they may not perform as well for a path with a large BDP [21].

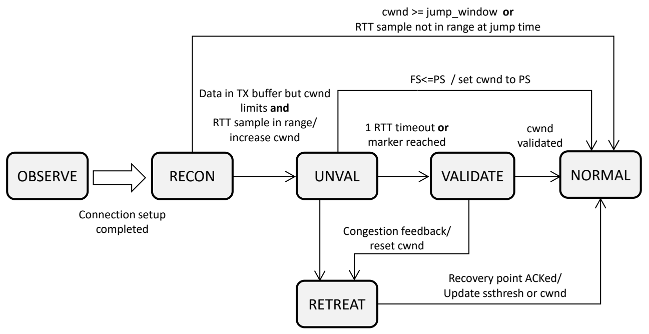

cwnd >= jump_window **or**
RTT sample not in range at jump time
FS<=PS / set cwnd to PS
Data in TX buffer but cwnd limits **and** RTT sample in range/ increase cwnd
1 RTT timeout **or** marker reached
cwnd validated
OBSERVE
RECON
UNVAL
VALIDATE
NORMAL
Connection setup completed
Congestion feedback/ reset cwnd
RETREAT
Recovery point ACKed/ Update ssthresh or cwnd

Fig. 1: Careful Resume state transition diagram

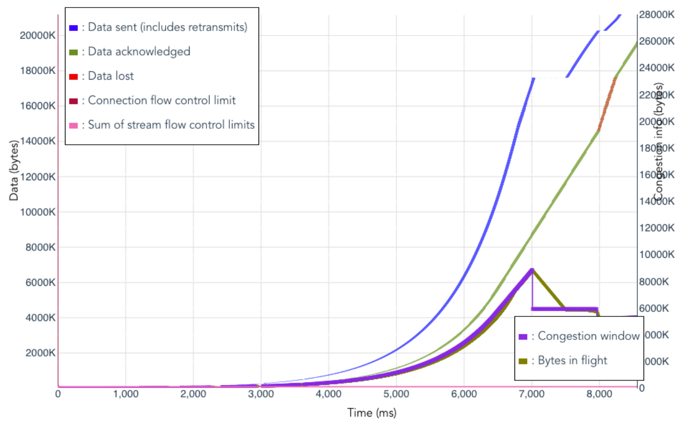
Several improvements have been proposed over the years to accelerate the performance of Slow Start. An IW of 10 maximum segment sizes was proposed in [22], increasing the data sent in the Slow Start phase. More recently, some proposals have emerged to expand IW to 64 packets (IW64). Additionally, more radical approaches, such as JumpStart [23], have proposed unbounded sizes of IW, but has not been deployed. An example is the HalfBack proposal [24], where IW is determined by the application, enabling even more aggressive utilisation of available capacity. Other proposals have also sought to adjust IW based on network feedback gathered over time in response to IW adjustments [13]. However, these approaches have had limited success due to the wide variability in application requirements and path characteristics.

For the large BDP encountered by satellite paths, an IW of 64 will be insufficient to saturate the path capacity. However, using larger IWs without additional information about the path incurs the risk of either overestimating or underestimating the capacity. Overestimating would result in overly conservative behaviour that fails to effectively utilise available capacity, while underestimating can lead to prolonged congestion periods, causing inefficiencies and degraded performance.

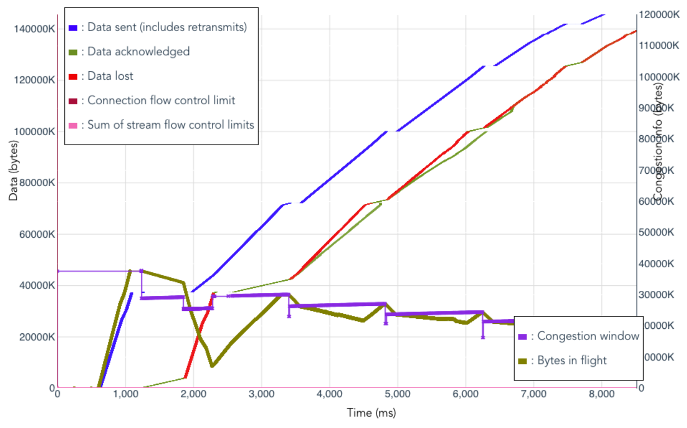
Another way to improve performance could be to reuse the CC parameters of a previous connection to initialise the CC parameters of a new connection. One approach is to leverage the TCP cache to improve performance, such as using Control Block Interdependence [13] to store the previous state of a connection, including parameters like the CWND and RTT. One such approach is Stateful TCP [14], which initialises the CWND based on the CWND of a previous connection. As highlighted in the design section, reusing previous states without an explicit validation phase can perform poorly when path or traffic conditions change significantly between connections to the same destination, or if the connection migrates to a different network.

III. Careful Resume Design

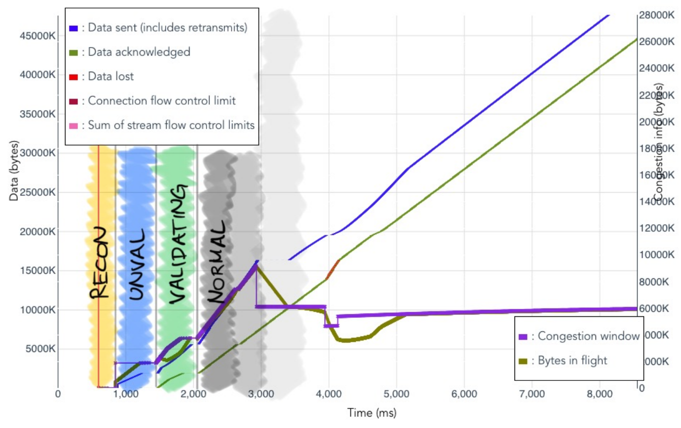
Similar to Stateful TCP, Careful Resume reduces the time to reach capacity after a connection restart by reusing capacity estimations from previous connections to the same destination.



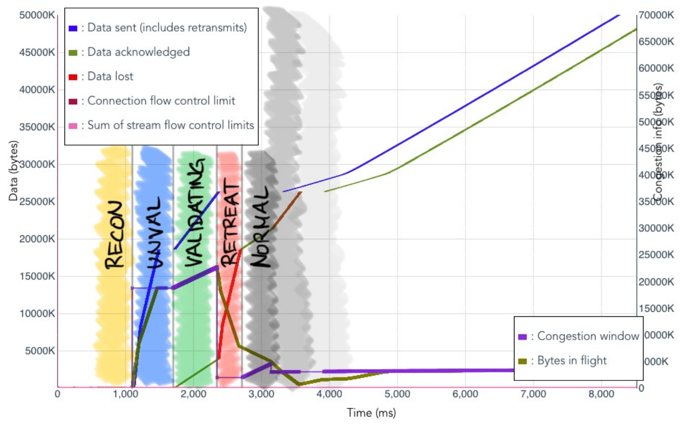
(a) HyStart



(b) Large Initial Window



(c) Careful Resume: Successfully increasing the CWND



(d) Careful Resume: Safe Retreat

Fig. 2: Dynamics of CC variables of QUIC connection on a GEO satellite path visualised with qvis [25]: In each subfigure, the left y-axis shows Data Sent/Acknowledged/Lost and flow control limits; the right y-axis shows the CWND and bytes in flight

Our approach includes an RTT validation phase, *Reconnaissance*, where a high transmission rate is only allowed if the current RTT is within a range of the previously measured minimum RTT. During high-rate transmission, if congestion is detected, a more vigorous rate reduction is applied than required for standard Slow Start. This is necessary because the congestion could result from changes in path characteristics or reduced capacity due to fading mitigation techniques when capacity is used without prior validation.

Figure 1 shows a state transition diagram of Careful Resume. A connection begins in the Reconnaissance state immediately after setup. If the RTT is validated and the transmission buffer holds sufficient data, the connection transitions into the *Unvalidated* phase, initiating transmission at a higher rate (half of the previously saved CWND). Packets sent in this phase are paced over one RTT. The reused CWND is based on capacity estimations. The draft does not specify how to measure the capacity, but the acknowledged bytes in one RTT could be measured. Congestion feedback from this phase is collected in a subsequent *Validating* phase during the following RTT.

Since connections start with $CWND = IW$, transfers smaller than an IW complete within 1 RTT after connection setup. Careful Resume is more significant for larger transfers, that can benefit from the jump window. Specifically, if the transfer size is less than half the BDP, the transfer ideally completes within one additional RTT. As the transfer size increases, Slow Start, in all cases, ultimately reaches a point where congestion is detected (with loss) and results in performance that is dominated by long-term, steady-state CC. Therefore Careful Resume most benefits medium-sized connections (e.g., web pages), where most of the data transfer is within the initial startup phase.

Any design of a CC algorithm needs to include a notion of *backoff* to rapidly reduce the sending rate to protect the network in times of extreme (persistent) congestion. Senders are encouraged to try to avoid causing excessively high rates of packet loss [15]. A key way to accomplish this is to avoid excessive increases in the sending rate, as in slow start. However, Careful Resume seeks to increase more rapidly, and therefore a sender needs to reduce its sending rate if it experiences high packet loss (e.g., overshoot after Careful

Resume performs the jump in the CWND into a bottleneck with less capacity than previously observed). This is the purpose of the Safe Retreat phase, where the response needs to be significantly greater than the response to steady state congestion.

If all packets sent during the *Unvalidated* phase are acknowledged, the connection moves to the *Normal* state, where standard CC rules apply.

A. Comparison of Slow Start / HyStart, Large Initial Window, and Careful Resume

Figure 2 compares the behaviour of HyStart, a very large Initial Window, and Careful Resume. These experiments have been obtained by an emulated geostationary satellite path ($600\,\mathrm{ms}$ RTT, $50\,\mathrm{Mbit/s}$) and Picoquic, which will be described in more detail in the following sections.

A typical Slow Start combined with HyStart is shown in Figure 2a. For each ACK, two new packets are sent, effectively doubling the CWND and sending rate each RTT. At $t = 7s$, HyStart is triggered because the RTTs have increased from $600\,\mathrm{ms}$ to approx. $700\,\mathrm{ms}$ due to buffering, the CWND is reduced by 0.7, and congestion avoidance is entered. The CWND and sending rate were already too high (almost $100\,\mathrm{Mbit/s}$) before the HyStart trigger. Thus, there is an overshoot and packets are lost, which can be seen after $t = 8s$. Without HyStart, Slow Start would have increased the CWND and overshoot even further, causing more packet loss and probably requiring multiple CWND reductions to get to or below the path capacity. The packet loss causes another reduction of the CWND, which is recovered shortly after, and the CWND and sending rate provide high path utilisation afterwards.

For Slow Start and HyStart, the number of RTTs to reach the path capacity is a significant performance metric. Careful Resume reduces this metric. Figure 2c illustrates how Careful Resume successfully increases the CWND. After the previous RTT is validated in the Reconnaissance phase, the CWND increases using pacing over the duration of one RTT. The jump to a high transmission rate in the Unvalidated phase is followed by the *Validating* phase, leading to a final transition to Normal. Because the CWND was set to only half of the previously estimated path capacity (allowing safe coexistence of other traffic), there is no packet loss. After transitioning from Validating to Normal, the CWND has grown up to the path capacity and normal CC operations continue.

Figure 2b demonstrates what happens if the IW would simply be set to very high values. In this example the IW is set to $10 \cdot BDP$, which is unrealistic for real deployments but suitable to illustrate the underlying problem. The transmission duration would actually be the smallest possible because new packets are always sent with a data rate exceeding the path capacity, but obviously, a large portion of packets from the too-large IW are lost. Afterwards, multiple rounds of congestion happen because CUBIC only reduces the CWND by a factor of 0.7 in each round. This results in a high number of losses and retransmissions again and again and would significantly harm other flows sharing a bottleneck.

If Careful Resume overshoots significantly, in the example shown in Figure 2d again by sending $10 \cdot BDP$ in the Unvalidated phase, the impact is less severe compared to a very large IW. Safe Retreat is entered as soon as packet loss (or a Congestion-Experienced signal) is detected in the Unvalidated or Validating phase. On entry to the Safe Retreat phase, the CWND is reduced to half of the measured *pipesize*, which drains any excessively sent packets from the path. The pipesize is given as the difference between packets sent in the Unvalidating phase and ACKs received at the sender during Unvalidating and Safe Retreat. The CWND is not increased during Safe Retreat. The transition to Normal is done when feedback for all packets sent during the Unvalidated phase has been received (either ACKed or lost). This typically happens within one RTT. After Safe Retreat, when transitioning to Normal, the Slow Start threshold (ssthresh) is updated to $pipesize \cdot Beta$, where Beta is a factor between 0.5 and 1. This means that normal CC operations are performed, which can be either Slow Start if CWND is smaller than ssthresh, or congestion avoidance.

For both Slow Start / HyStart and Careful Resume, the long-term behaviour is determined by the CC algorithm; Careful Resume only impacts the start of connections.

IV. Implementation of Careful Resume

Our implementation of Careful Resume is based on Christian Huitema's *Picoquic* [26], an open-source QUIC server and client written in C, widely used in research. While Picoquic supports several CC algorithms, including CUBIC [27], NewReno [28], and BBR [20], this paper focuses on CUBIC.

CC code in Picoquic is organised in separate modules facilitating development and testing. A CC algorithm is implemented via a callback function triggered by a specific event, such as ACK reception, packet loss detection, or an RTT measurement. The callback function handles all events, with arguments that include the connection context and the type of event notification.

In our implementation of Careful Resume, which is available online,[1] handling of CC notifications and phase transitions are consistent across all CC algorithms. Only minor modifications in Picoquic's CUBIC implementation were required to integrate Careful Resume. It would, therefore, be straightforward to upstream Careful Resume into the official Picoquic repository.

Our implementation includes QLOG support for Careful Resume events, as described in [12]. QLOG allows the use of graphical tools like *qvis* [25] to analyse CC behaviour, as demonstrated in this paper.

Picoquic comes with a simplified built-in discrete event simulator used for testing. We added several Careful Resume tests, available online.[2]

[1]https://github.com/hfstco/picoquic/tree/cr

[2]https://github.com/hfstco/picoquic/blob/cr/picoquictest/careful_resume_test.c

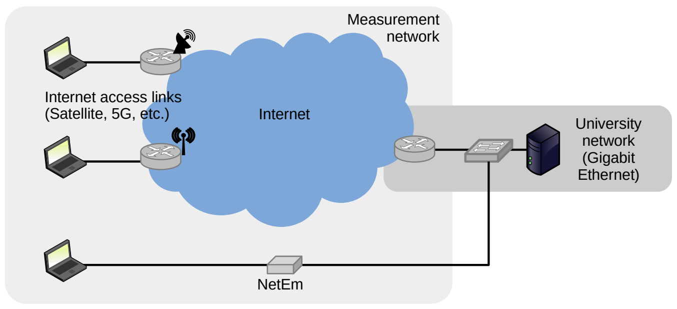

Measurement network
Internet access links (Satellite, 5G, etc.)
Internet
University network (Gigabit Ethernet)
NetEm

Fig. 3: Measurement setup.

V. Testbed Setup

Careful Resume was evaluated across multiple Internet access technologies using the testbed shown in Figure 3. A Picoquic client ran on a computer (Intel Core i5, 16 GB RAM) connected to either a NetEm[3] link impairment computer, or the modem/router provided by the operator connected to the operator network and Internet. The Picoquic server (Intel Xeon X5690, 128 GB RAM) was connected via Gigabit-Ethernet to the University network [29]. Table I summarises the hardware, download (forward link) and upload (return link) data rates, and typical RTTs.

Our evaluations focus on paths with a large BDP, especially satellite Internet access. As a baseline and comparison with real GEO satellite paths, NetEm was configured with a delay of $300\,\mathrm{ms}$ in each direction, a rate of $50\,\mathrm{Mbit/s}$ for the forward link (download) and $5\,\mathrm{Mbit/s}$ for the return link (upload), and a bottleneck buffer size of one BDP ($3.75\,\mathrm{Mbyte}$ in the forward link). The performance of a CC algorithm is affected by the queueing discipline at the bottleneck and we consider a drop-tail queue discipline (using a FIFO buffer).

Three GEO Internet access providers (Konnect[4], SkyDSL[5], and Brdy[6]) deploy split-TCP PEPs [5] to enhance TCP connection performance. These PEPs cannot be applied at the transport layer when protocol headers are encrypted, as is the case with Virtual Private Networks (VPNs) or when using QUIC [7], [30].

Another test was conducted over Starlink [31], a satellite mega-constellation in Low Earth Orbit (LEO). Although Starlink offers higher rates and significantly lower RTTs than GEO Internet access, current services may still experience packet loss [31], [32].

Additionally, a terrestrial 5G fixed wireless Internet access (Telekom MagentaMobil Prepaid Max) was set up to evaluate the performance of Careful Resume with cellular paths.

VI. Results

This section compares two RTT metrics: the current RTT and the minimum RTT, to understand which to choose for Careful Resume. Next, the performance of Careful Resume is evaluated for different path characteristics.

A. *Reconnaissance*

RTT validation occurs during the Reconnaissance phase to determine whether the measured RTT aligns with the previously observed value. If the values do not match, the attempt to switch to a higher rate is abandoned, and the normal startup continues. Changes in RTT can occur after events like connection migration, path changes, or mobile network handovers, where significant variations in delay are detectable.

To enhance the accuracy of path RTT measurements at startup and improve correct path identification, multiple RTT samples from within IW could be used. However, this must be weighed against the server's delay in collecting this data, which could postpone rate increases. Indeed, QUIC packets may be paced within IW, potentially introducing delays of up to one RTT as packets are spread over that duration.

[3]https://wiki.linuxfoundation.org/networking/netem
[4]https://europe.konnect.com
[5]https://www.skydsl.eu
[6]https://brdy.com

TABLE I: Testbed Internet Providers. Except for NetEm, data rates and RTTs are approximate values.

	Hardware	Download	Upload	RTT
NetEm		50 Mbit/s	5 Mbit/s	600 ms
Konnect 7.2°E	Hughes WR3200	50 Mbit/s	5 Mbit/s	600 ms
SkyDSL 9°E	ViaSat RM5111	50 Mbit/s	6 Mbit/s	600 ms
Brdy 9°E	ViaSat RM4100	25 Mbit/s	6 Mbit/s	600 ms
Starlink	Rev. 1 dish UTR-201 router	up to 480 Mbit/s	up to 100 Mbit/s	40 ms
Telekom 5G	Huawei 5G CPE Pro 2	up to 500 Mbit/s	up to 50 Mbit/s	30 ms

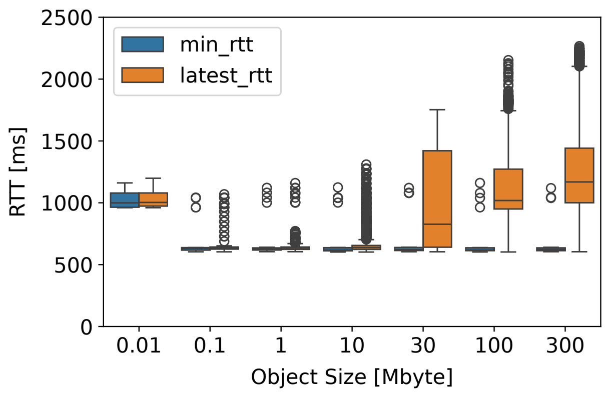
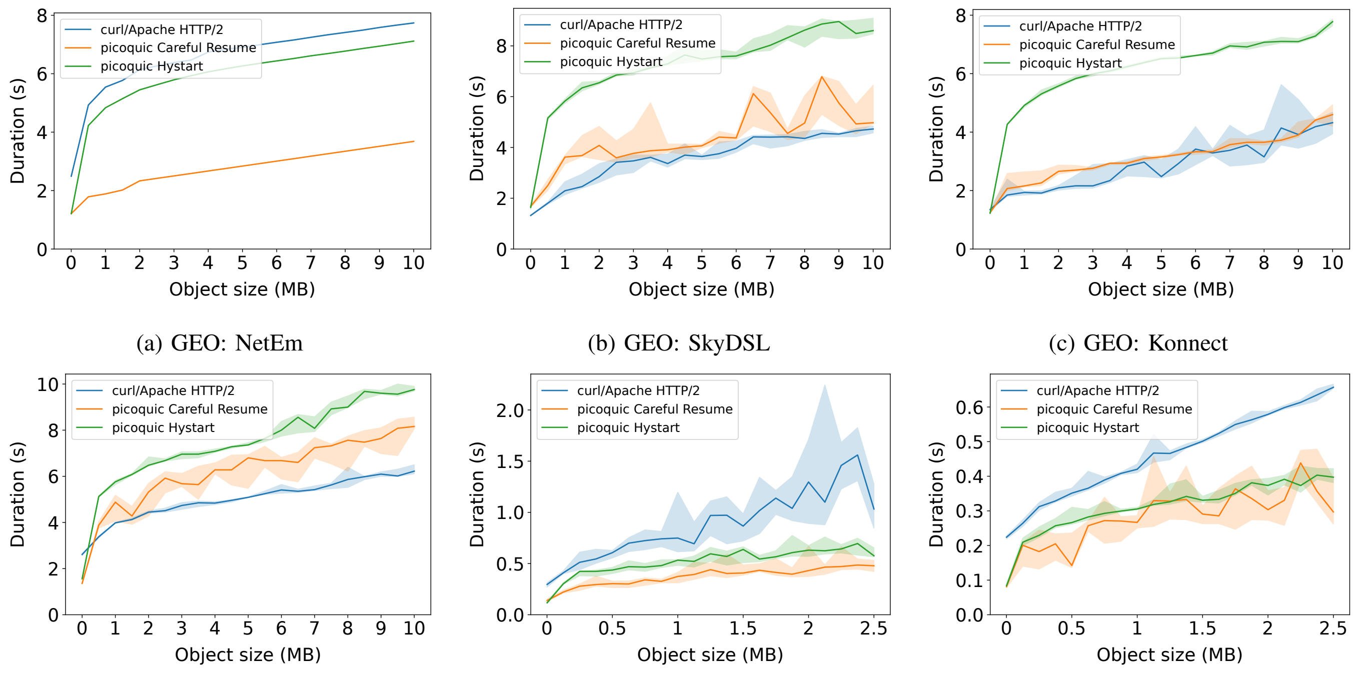


Fig. 4: Current and minimum RTTs for different object sizes. Each object was transferred 10 times. The larger the object, the more buffering, and thus higher RTTs can be observed. Measurements were taken using a real GEO satellite operator (SkyDSL) and Picoquic with CUBIC congestion control. Other Internet paths show similar behaviour.



(a) GEO: NetEm (b) GEO: SkyDSL (c) GEO: Konnect

(d) GEO: Brdy KA-SAT (e) LEO: Starlink (f) 5G cellular: Telekom 5G

Fig. 5: HTTP single object transfer duration. Solid lines show the median values, shaded areas the 1st and 3rd quartiles.

When the forward path uses a capacity management/allocation method, the available resource is periodically distributed among the active nodes. This results in a varying path delay, which is common for both GEO and LEO satellite Internet services and 3GPP mobile networks. In these cases, the delay varies as a function of external factors, and attempting to infer congestion from an increase in the delay results in reduced throughput. This variation in the delay over short timescales (jitter) in these paths are not distinguishable from jitter that results from other (radio adaption) effects.

To mitigate initial delays and jitter, our implementation limits RTT sampling to a single comparison against the previously measured RTT. Figure 4 shows that for a GEO satellite path, the minimum RTT distribution is stable compared to the average RTT over a connection's duration. In satellite connections, the average and variance of RTT generally increase over the session, while the minimum RTT remains stable and may decrease as variance reduces. The minimum RTT is chosen for evaluation, and a range covering 100% of the variance is used for validation, considering buffer occupancy and possible fluctuations within a one-to-two RTT range.

B. *Performance evaluation*

Careful Resume enables connections to achieve their fair share of bandwidth faster than Slow Start, reducing overall transfer time. To demonstrate the benefits of Careful Resume, we measured transfer times across a range of object sizes. Data requests were made over HTTP/3 from client to server, with 10 iterations per object size. All paths except the NetEm setup traverse the public Internet, for which we expect a higher variability compared to NetEm.

Because the Careful Resume specification does not specify the *Observe* phase in detail, the `saved_cwnd` is set to one BDP and the `saved_rtt` is set to the value listed in Table I.

Figure 5 shows the transfer times for object sizes between 1 Kbyte and 10 Mbyte with a step size of 500 Kbyte, comparing three different HTTP transfers:

- *Picoquic HyStart (HTTP/3+QUIC)*: A version with no additional mechanisms beyond the standard for CUBIC. It implements HyStart, as described in [16], and follows the CUBIC specifications in RFC 9438. We purposely removed Picoquic's updating of the CWND based on bandwidth measurements[7] as this is not standard-compliant and would not allow a fair comparison with Careful Resume.
- *Picoquic Careful Resume (HTTP/3+QUIC)*: Picoquic equipped with Careful Resume, where the connection starts with half the previously measured CWND, reflecting a typical restart scenario.
- *curl/Apache (HTTP/2+TCP)*: Data is transferred via HTTP/2+TCP to compare our results with the mature and widely deployed tools *curl* and *Apache*. Unlike QUIC, PEPs can be applied to TCP traffic.

Figure 5a shows that an emulated GEO path, where no PEP is used, has very high transmission times for HTTP/2+TCP and HTTP/3+QUIC unless Careful Resume is used. Without a PEP configuration to support HTTP/2 on the emulated satellite

[7]https://github.com/private-octopus/picoquic/blob/ab97f070143cd7133ee108e1ea549f36f1d17eba/picoquic/cubic.c#L206-L214

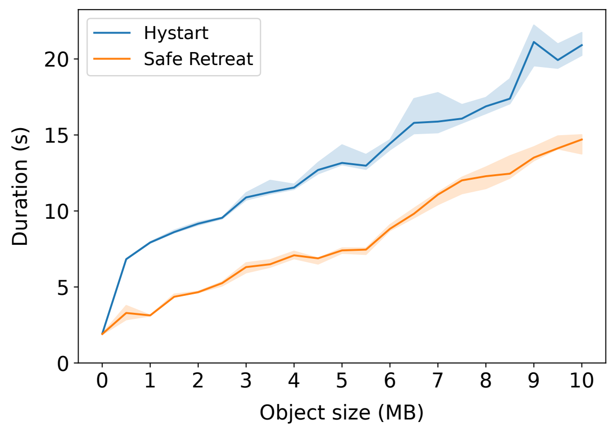


Fig. 6: HTTP single object transfer duration of HyStart vs. CR when starting in the presence of another (ongoing) flow. Experiments were performed using NetEm (see Table I) and repeated 10 times. Solid lines show the median values, shaded areas the 1st and 3rd quartiles.

link testbed, HTTP/2 experiences significant delays, leading to performance comparable to that of Slow Start.

The benefit of PEPs can be seen in Figure 5b, 5c, and 5d, where HTTP/2+TCP performs significantly better than QUIC without Careful Resume. In most cases, QUIC with Careful Resume performs almost as well as HTTP/2 with PEP acceleration. QUIC with Careful Resume eliminates a significant portion of the time spent in Slow Start and offers an advantage of approximately two seconds compared to the performance of *Picoquic HyStart*, equivalent to about 3 to 4 RTTs.

Starlink, shown in Figure 5e, provides much smaller RTTs than GEO satellite Internet providers. Overall, the performance of HTTP/3+QUIC is better than HTTP/2+TCP, and Careful Resume further reduces the transmission time, again by 3 to 4 RTTs.

For 5G, the transmission times for HTTP/2 and Careful Resume are nearly the same; therefore, Careful Resume did not provide a benefit but did not cause any performance penalties. Fluctuations in the data rate make it difficult for Careful Resume to successfully jump to higher CWND values. Many connections transition into Safe Retreat, which explains the noisy results for Careful Resume. These results indicate that the method is safe for a 5G path, but that it is likely to result in less benefit compared to a satellite path.

C. Safe Retreat and Performance with Crosstraffic

The Safe Retreat phase is an essential part of Careful Resume, designed to avoid persistent congestion following a significant overshoot and to prevent rate increases from negatively impacting other flows. When congestion is detected in either the Unvalidated or Validating phase, Careful Resume immediately reduces the CWND and drains excess unvalidated packets from the path, converging within one RTT. Even if Careful Resume enters Safe Retreat, the transfer time might still be reduced because the first part of the transfer may already have been sent in the Unvalidated phase.

In analysing a new CC mechanism, it is important to also consider the effect on other flows that might share the bottleneck bandwidth. This could be the case, for instance if another connection starts using the path before the connection using Careful Resume is started.

Figure 6 shows a comparison of HyStart and Safe Retreat, when the path is already occupied by another competing CUBIC flow. The competing flow is in steady-state (congestion avoidance or recovery), and the bottleneck buffer is partially filled. We consider a path with a GEO satellite, and a bottleneck capacity of 1 BDP.

When a new HyStart flow enters the bottleneck, it experiences increased RTTs or packet loss early because of the competing flow and already occupied buffer and transitions to congestion avoidance. It thus achieves a small CWND with a longer transmission duration, seen when comparing HyStart without Crosstraffic (Figure 5a) and HyStart with Crosstraffic (Figure 6). This is expected, and not related to Careful Resume.

Suppose a new flow using Careful Resume starts sending and successfully validates the path in the Reconnaissance phase, because RTTs are similar to the previously saved values. It then increases the CWND in the Unvalidated phase, but then experiences packet loss in the Validating phase, with a transition to Safe Retreat. After Safe Retreat, upon entering the Normal phase, the flow continues to perform using normal CUBIC congestion control, just like HyStart. As expected, the performance of Careful Resume undergoing Safe Retreat (Figure 6) is worse than Careful Resume with a successful jump (Figure 5a). Still, even if Careful Resume transitions to Safe Retreat, part of the data was transmitted and the pipesize measured, which is a benefit compared to Slow Start and HyStart. This results in a lower transmission duration for Careful Resume with Safe Retreat compared to HyStart, as shown in Figure 6.

VII. Conclusion and Future Work

Slow Start or HyStart can take multiple RTTs to reach the path capacity, while still causing overshoot and packet loss. This paper describes a new method, Careful Resume, that improves this by reusing CC parameters from previous connections and several safety measures, i.e., the Reconnaissance and Safe Retreat phase. We have presented the design and benefits of Careful Resume and first performance evaluations.

We integrated Careful Resume into Picoquic without major code changes, and our implementation is publicly available. We focused on CUBIC which is currently the most widely deployed CC algorithm. Although we do not present data in this paper, we suggest the approach is also applicable to other CC algorithms, including BBR.

We present the first published analysis of the new Careful Resume approach. Our analysis used NetEm and real Internet paths for the performance evaluation, including three GEO satellite providers, the LEO Starlink network, and a cellular

5G provider. Significant improvements were observed for high BDP paths, such as GEO satellite paths, where Careful Resume can provide a solution to the non-applicability of PEPs. Performance gains were observed for the LEO Starlink and cellular 5G network when HTTP/3+QUIC is used instead of HTTP/2+TCP; Careful Resume provided some additional gains. As future work, testing further providers, paths, and vantage points is foreseen.

Preventing congestion collapse and doing no harm to other flows are the fundamental design principles of CC. Careful Resume realises this by design and safety measures, and we have evaluated Careful Resume when starting in the presence of an ongoing flow. However, our evaluation was limited to a single competing flow, which should be analysed in more detail with multiple competing flows, including Jain's fairness index and the effect of fair queuing.

Acknowledgments

The contents presented are part of the outcomes of the European Space Agency (ESA) QUICOPTSAT project, see https://connectivity.esa.int/projects/quicoptsat, contract 4000139035/22/NL/AF. The authors would like to thank ESA for the project. Responsibility for the contents resides with the authors.

References

[1] J. Iyengar and M. Thomson, "QUIC: A UDP-Based Multiplexed and Secure Transport," RFC 9000, May 2021. [Online]. Available: https://www.rfc-editor.org/info/rfc9000

[2] A. Langley and et al., "The QUIC Transport Protocol: Design and Internet-Scale Deployment," *Proc. of SIGCOMM*, 2017. [Online]. Available: https://api.semanticscholar.org/CorpusID:2768765

[3] E. Rescorla, M. Thomson, and K. Oku, "Using TLS to Secure QUIC," RFC 9001, May 2021. [Online]. Available: https://www.rfc-editor.org/info/rfc9001

[4] G. Fairhurst and C. Perkins, "RFC 9065: Considerations around Transport Header Confidentiality, Network Operations, and the Evolution of Internet Transport Protocols," USA, 2021.

[5] J. Griner, J. Border, M. Kojo, Z. D. Shelby, and G. Montenegro, "Performance Enhancing Proxies Intended to Mitigate Link-Related Degradations," RFC 3135, Jun. 2001. [Online]. Available: https://www.rfc-editor.org/info/rfc3135

[6] T. Pauly, E. Rosenberg, and D. Schinazi, "QUIC-Aware Proxying Using HTTP," IETF, Internet-Draft draft-ietf-masque-quic-proxy-04, Oct. 2024, work in Progress. [Online]. Available: https://datatracker.ietf.org/doc/draft-ietf-masque-quic-proxy/04/

[7] L. Thomas, E. Dubois, N. Kuhn, and E. Lochin, "Google QUIC performance over a public SATCOM access," *International Journal of Satellite Communications and Networking*, vol. 37, no. 6, pp. 601–611, 2019. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/sat.1301

[8] J. Deutschmann, K.-S. Hielscher, and R. German, "Satellite Internet Performance Measurements," in *2019 International Conference on Networked Systems (NetSys)*, 2019, pp. 1–4.

[9] A. Custura, T. Jones, and 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.

[10] J. Border, B. Shah, C.-J. Su, and R. Torres, "Evaluating QUIC's Performance Against Performance Enhancing Proxy over Satellite Link," in *2020 IFIP Networking Conference (Networking)*, 2020, pp. 755–760.

[11] S. Endres, J. Deutschmann, K.-S. Hielscher, and R. German, "Performance of QUIC Implementations Over Geostationary Satellite Links," 2022. [Online]. Available: https://arxiv.org/abs/2202.08228

[12] N. Kuhn, E. Stephan, G. Fairhurst, R. Secchi, and C. Huitema, "Convergence of Congestion Control from Retained State," IETF, Internet-Draft draft-ietf-tsvwg-careful-resume-10, Jul. 2024, work in Progress. [Online]. Available: https://datatracker.ietf.org/doc/draft-ietf-tsvwg-careful-resume/10/

[13] D. J. D. Touch, M. Welzl, and S. Islam, "TCP Control Block Interdependence," RFC 9040, Jul. 2021. [Online]. Available: https://www.rfc-editor.org/info/rfc9040

[14] L. Guo and J. Y. B. Lee, "Stateful-TCP—A New Approach to Accelerate TCP Slow-Start," *IEEE Access*, vol. 8, pp. 195 955–195 970, 2020.

[15] S. Floyd and M. Allman, "Specifying New Congestion Control Algorithms," RFC 5033, Aug. 2007. [Online]. Available: https://www.rfc-editor.org/info/rfc5033

[16] S. Ha and I. Rhee, "Taming the elephants: New TCP slow start," *Computer Networks*, vol. 55, no. 9, pp. 2092–2110, 2011. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1389128611000363

[17] E. Blanton, D. V. Paxson, and M. Allman, "TCP Congestion Control," RFC 5681, Sep. 2009. [Online]. Available: https://www.rfc-editor.org/info/rfc5681

[18] S. Floyd, "Limited Slow-Start for TCP with Large Congestion Windows," RFC 3742, Mar. 2004. [Online]. Available: https://www.rfc-editor.org/info/rfc3742

[19] P. Balasubramanian, Y. Huang, and M. Olson, "HyStart++: Modified Slow Start for TCP," RFC 9406, May 2023. [Online]. Available: https://www.rfc-editor.org/info/rfc9406

[20] N. Cardwell, I. Swett, and J. Beshay, "BBR Congestion Control," IETF, Internet-Draft draft-cardwell-ccwg-bbr-00, Jul. 2024, work in Progress. [Online]. Available: https://datatracker.ietf.org/doc/draft-cardwell-ccwg-bbr/00/

[21] J. Deutschmann, K.-S. Hielscher, and R. German, "CUBIC Local Loss Recovery vs. BBR on (Satellite) Internet Paths," in *In proc of LANMAN*, 2023, pp. 1–3.

[22] J. Chu, N. Dukkipati, Y. Cheng, and M. Mathis, "Increasing TCP's Initial Window," RFC 6928, Apr. 2013. [Online]. Available: https://www.rfc-editor.org/info/rfc6928

[23] D. Liu, M. Allman, S. Jin, and L. Wang, "Congestion Control Without a Startup Phase," in *Proc. PFLDnet*, 2007.

[24] Q. Li, M. Dong, and P. B. Godfrey, "Halfback: running short flows quickly and safely," in *Proc. of ACM ConNext*, ser. CoNEXT '15. New York, NY, USA: ACM, 2015. [Online]. Available: https://doi.org/10.1145/2716281.2836107

[25] Robin Marx, "qvis, the QUIC and HTTP/3 visualization toolsuite," https://qvis.quictools.info/.

[26] C. Huitema, "picoquic," https://github.com/private-octopus/picoquic, 2017.

[27] L. Xu, S. Ha, I. Rhee, V. Goel, and L. Eggert, "CUBIC for Fast and Long-Distance Networks," RFC 9438, Aug. 2023. [Online]. Available: https://www.rfc-editor.org/info/rfc9438

[28] A. Gurtov, T. Henderson, S. Floyd, and Y. Nishida, "The NewReno Modification to TCP's Fast Recovery Algorithm," RFC 6582, Apr. 2012. [Online]. Available: https://www.rfc-editor.org/info/rfc6582

[29] DFN (Deutsches Forschungsnetz), "The national research and education network," https://www.dfn.de/en/network.

[30] J. Deutschmann, S. Jahandar, K.-S. Hielscher, and R. German, "Internet via Satellite: GEO vs. LEO, OpenVPN vs. Wireguard, and CUBIC vs. BBR," in *Proc. of MobiCom Workshop on Satellite Networking and Computing*. New York: ACM, 2023, p. 19–24.

[31] M. Bülo, J. Deutschmann, K.-S. Hielscher, and R. German, "Crowdsourced Starlink Performance Measurements from https://starlinkstatus.space," in *ICC 2024 - IEEE International Conference on Communications*, 2024, pp. 5596–5603.

[32] J. Deutschmann, K.-S. Hielscher, and R. German, "Broadband Internet Access via Satellite: Performance Measurements with different Operators and Applications," in *Broadband Coverage in Germany; 16th ITG-Symposium*, 2022, pp. 1–7.