i2cat R

Research & Innovation Centre

Deploying QUIC in Challenged Networks

Profiling and Volume-Aware Congestion Control

Oriol Fusté[1,2] · Juan A. Fraire[3,4]

Anna Calveras Augé[1] · Joan A. Ruiz-de-Azua[1,2]

[1]UPC BarcelonaTech · [2]i2CAT · [3]Inria/INSA Lyon · [4]CONICET–UNC

STINT 2026

CERCA Centres de Recerca de Catalunya

ACCIÓ Catalonia Trade & Investment · Generalitat de Catalunya

tecnio catalonia

Vigència: 29/12/2028

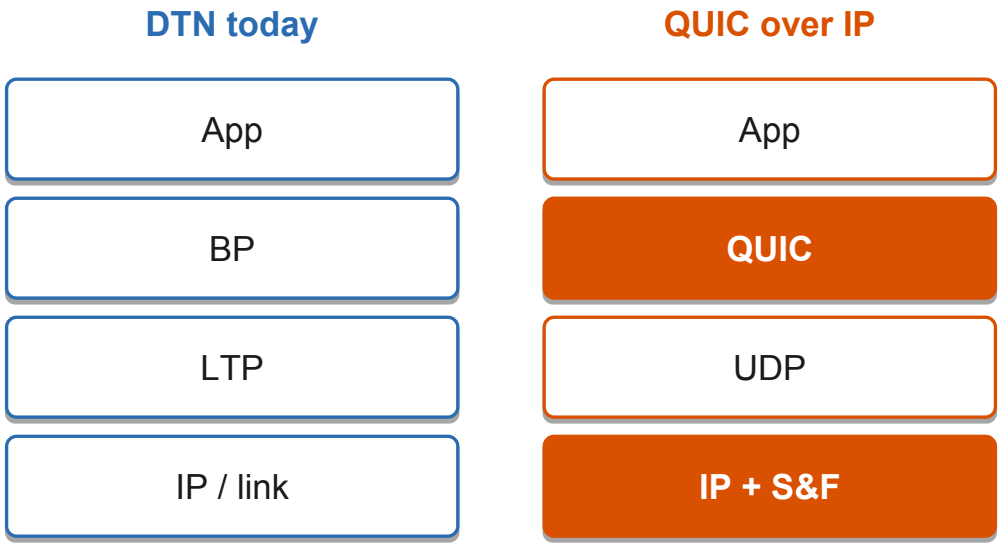
The default answer, and the new alternative

- **DTN:** BPv7, in-network store & forward, late binding, flight-proven, standard in CCSDS / LunaNet
- **2024** → IETF draft revisits discarding IP for deep space:
  - *“TCP’s flaws are TCP’s, not IP’s”*

QUIC is modern and flexible:

- ACK-based loss detection, no retransmission ambiguity
- Runs in user space
- Configurable time constants
- Exchangeable Congestion Control Algorithm (CCA)

→ **“tiptop” WG** aims for a QUIC/IP stack for deep space



One question at a time

The “Interplanetary Internet” has **two** problems combined:

A challenged network
extreme delay · disruption · no continuous path

+ a network of networks
heterogeneous links, segments, rates and administrations

Most of the literature studies **specific scenarios**, so both problems manifest at once, blurring more fundamental analysis.

→ Before focusing on a scenario or comparing between stacks, we should answer some more general and fundamental questions

Does end-to-end QUIC work in a challenged network at all?

What the literature settles

The literature agrees: QUIC parameters adjusted according to topology dynamics keep the connection alive through scheduled disruptions. → **Profiling works**

Congestion control is a different issue:

- Some **disable CC** or traffic is application limited, reducing its applicability to real networks
- Others test **terrestrial CCAs** (loss/delay-based) and **underperform**

– M. Blanchet and W. M. Eddy, “Internet Protocol Stack in Deep Space: Architecture and Simulation Results,” in Proc. 76th International Astronautical Congress (IAC), (Sydney, Australia), Sept. 2025
– Johannes Frisch, “Evaluation of QUIC in Interplanetary Networks,” Master’s thesis, Technische Universit¨at Darmstadt, Apr. 2026.
– C. Caini and T. d. Cola, “Earth to Moon Networking: Comparison of End-to-End and DTN approaches,” IEEE Aerospace and Electronic Systems Magazine, pp. 1–10, 2026.
– Github: Deepspaceip/deepspace-ip-simulations, Internet Protocol Suite for Deep Space Networking, Jul. 2026.

This work tries to answer two questions

1. Is profiling alone enough for predictable challenged networks?

2. Can open-loop congestion control be an alternative to classic CCAs?

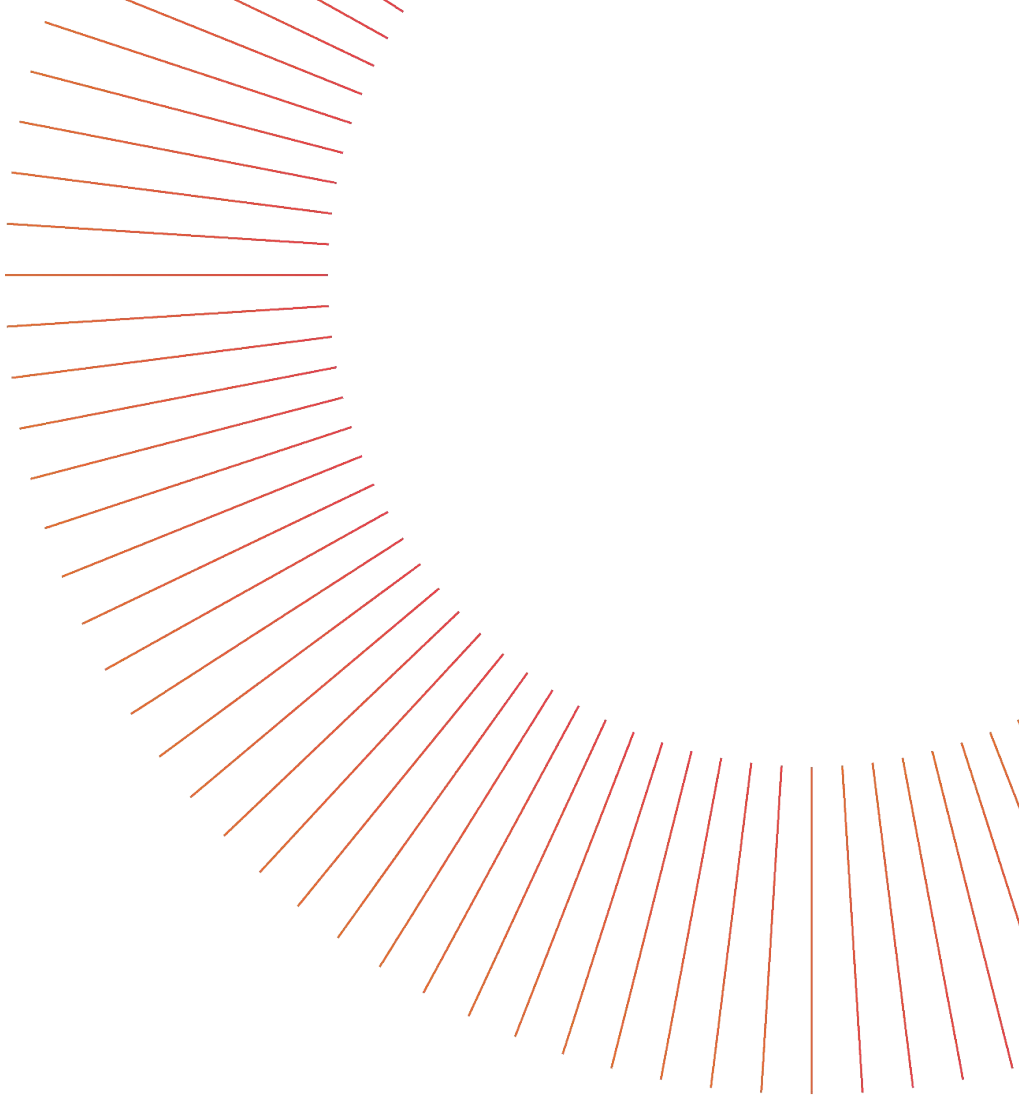
DISCLAIMER

The authors:

- DO NOT CLAIM this work to be directly transferable to Deep Space/IPN discussions
- DO NOT CLAIM that a QUIC/IP stack is better than any other, we only explore if it could work

Congestion Control

Why the classic signals fail — and what replaces them



WHAT WE PROPOSE

Congestion control: three problems

Classic closed-loop CCAs have three weaknesses in challenged networks:

1. Time
feedback arrives *after* the network has already changed

2. Signaling
loss ≠ congestion · RTT increase ≠ congestion → Network dynamics mislead the algorithm

3. Target value — BDP
Tries to find an optimal value under a wrong assumption

Baked into every terrestrial CCA logic — this is not a tuning problem!

i2cat

An open-loop alternative

- We need to stop trying to react to the network, we need to be **proactive**
- We need to predict the network → **Contact Plan**
  - It’s the same information leveraged by DTN and is also be used for routing

But how does the contact plan tell us how much we can send?

- Not rate × time (BDP) — the path is not a static pipe.
- We need a different reference quantity!

There is no pipe — there is a chain of buckets

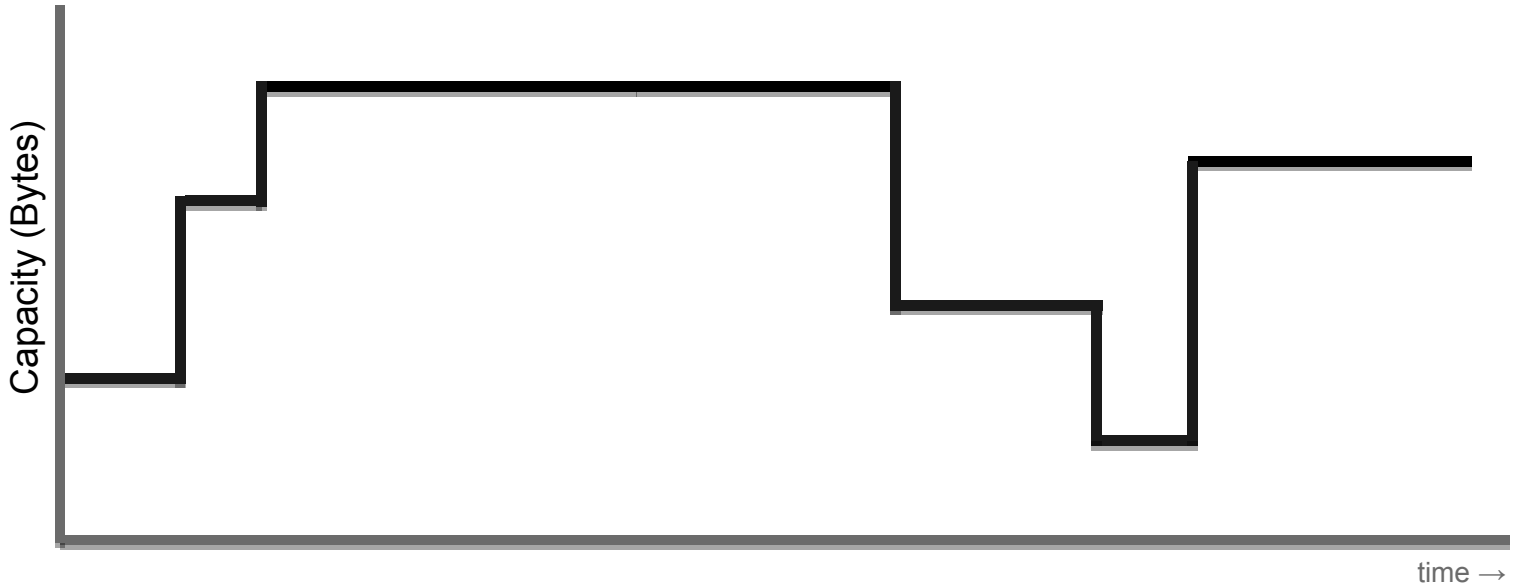
In a challenged network, congestion is S&F storage exhaustion — not router buffer overflow.

the model classic CCA assumes

bandwidth × RTT

one rate · one continuous path

what a challenged network actually offers



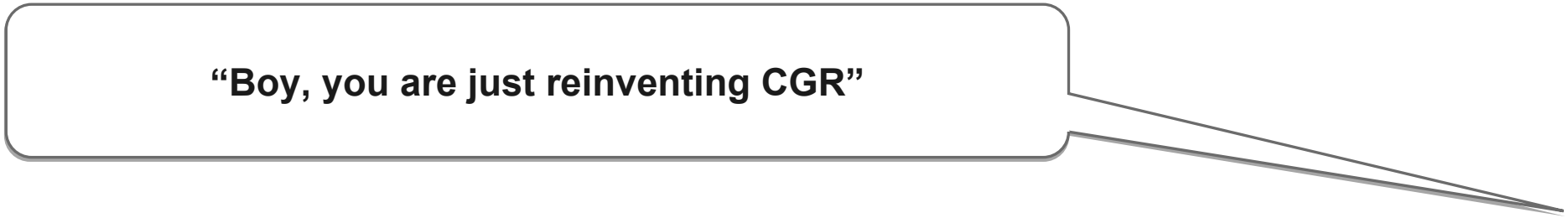
Path capacity

Over one path opportunity, what the sender can push end to end is a **volume**, not a rate:

$$V\_path = \min_i \left(R_i \times T_i^{eff} , \; storage_i \right)$$

- R_i, T_i^{eff} — each hop's contact rate and *effective* duration along the S&F path
- **$storage_i$** — each node's buffer along that same path
- V_path is the **minimum** of all of them — the tightest hop or smallest buffer sets the volume for the whole opportunity

Relation to DTN: same rationale, different layer structure



DTN/CGR	volume-aware CCA
per-bundle forwarding decisions	**one scalar window** per QUIC connection
evaluated at **every hop**, against the full contact graph	evaluated **at the source only**, with no routing authority
Decisions reinforced with local **live information**	**Window unchanged** by live events

→ The **rationale** transfers, the **mechanism** does not

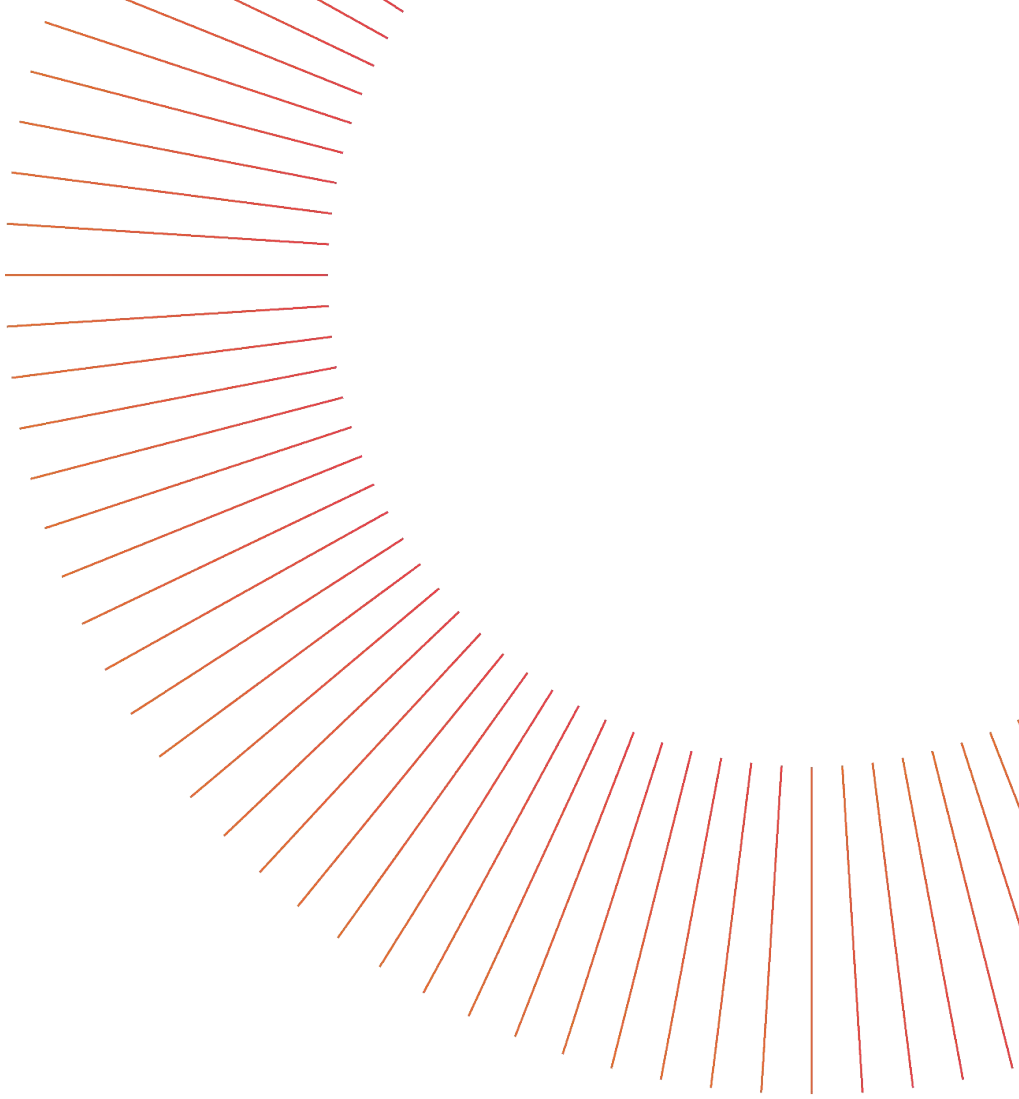
Sizing the congestion window to path capacity

Proposed “volume-aware” CCA:

- **Lock** the congestion window to the current path’s routable capacity (**scheduled CWND**)
- **Ignore loss and delay** entirely; ACKs only *clear* bytes
- Update the window as soon as the sender finishes the preceding patht’s contact

Profiling

Every QUIC parameter from the contact plan



WHAT WE PROPOSE

QUIC profiling: from tuning to derivation

Parameter	Derived from	Failure if wrong
`kInitialRtt`	> expected handshake RTT	premature PTOs
`IdleTimeout`	> longest scheduled disruption	connection dropped mid-downtime
`max_ack_delay – ACK frequency`	largest allowed — 10	↓ responsiveness on lossy links
`kPacketThreshold` / `kTimeThreshold`	Adapt to worst-case expected reorder	spurious loss declaration
`kInitialWindow`	first path's routable volume	under → contact wasted; over → S&F overrun

Other settings: 0-RTT with pre-shared keys, flow control deactivated

All can be derived from the contact plan and node characteristics.

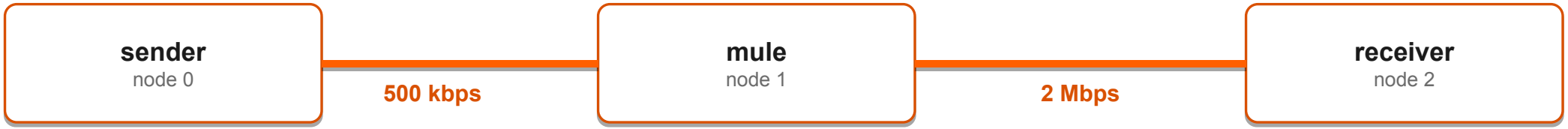
i2cat

Evaluation: a generalised data mule

- 3 IPv6 S&F nodes
- Limited Storage, contacts never overlap
- Path volumes span **18.75 → 150 MB** over ~53 h — deliberately *dynamic*
- Sender always has data; receiver is a sink — Never Application limited
- ns-3 simulation

Three stacks compared

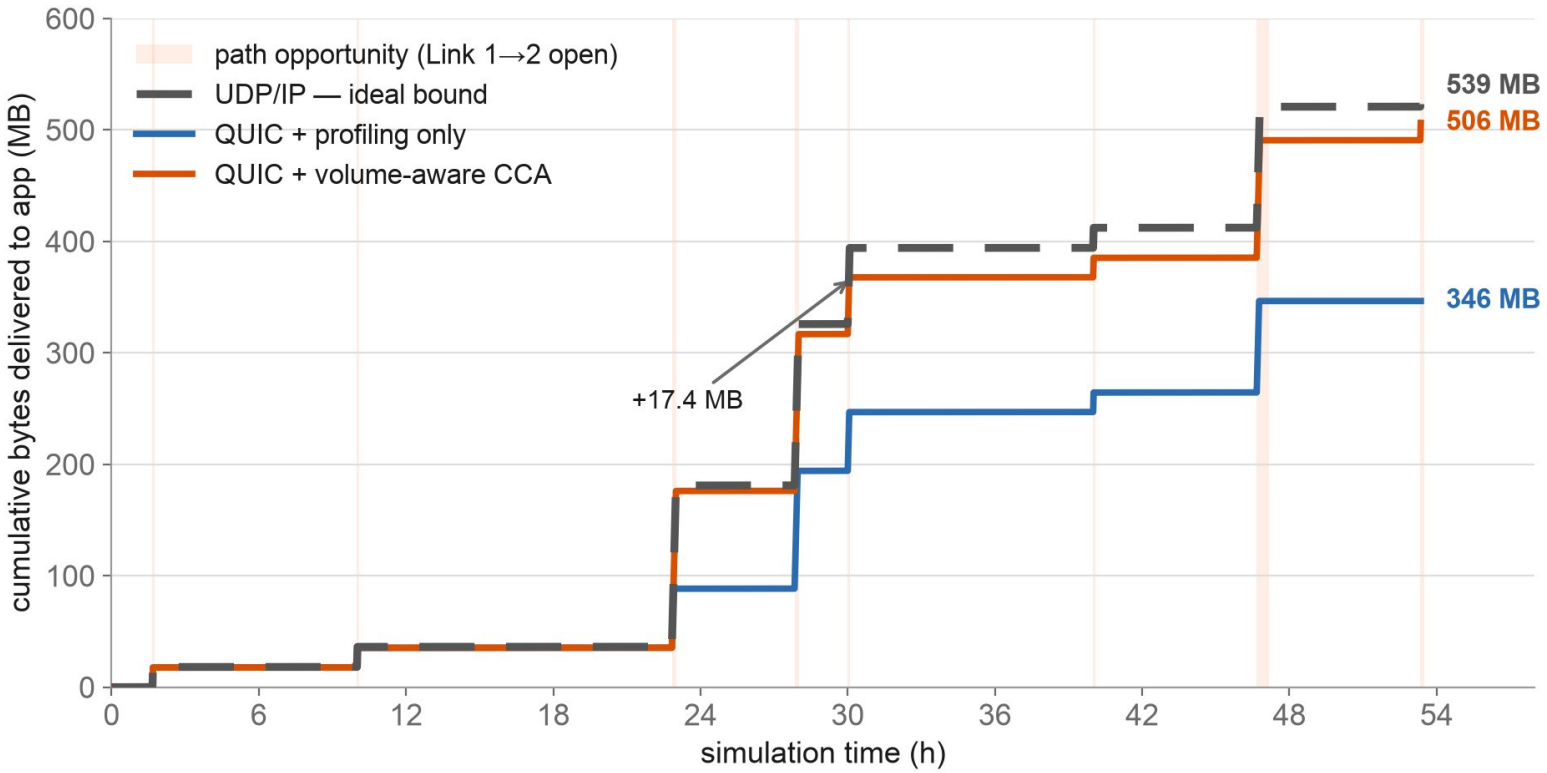
1. **UDP/IP** — ideal bound, no CC
2. **QUIC + profiling only** (default New Reno)
3. “ ” **+ volume-aware CCA**



ns-3 QUIC implementation

- *Mostly* compliant with RFC 9000 & 9002
- Fully configurable QUIC specification parmeters
- No TLS security implemented
- Forked from abandoned code at SIGNET Lab - DEI, University of Padova
- Openly available at: https://github.com/ENTEL-WNG/quic-ns-3

Result 1: profiling alone is not enough



UDP → 539 MB

the bound — but with **no CC at all**

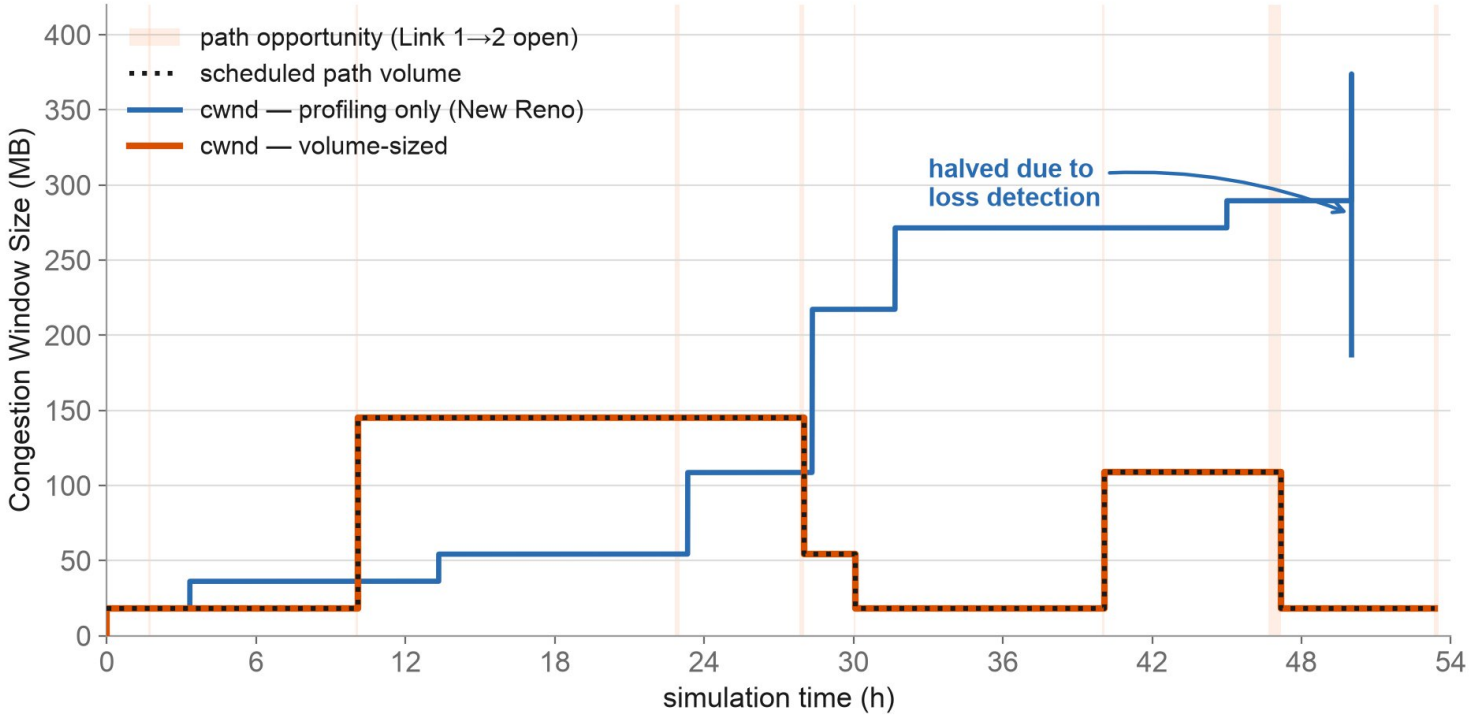
Profiling only → 346 MB

64 % of the bound

Volume-Aware CCA → 506 MB

93% of the bound

Result 1b: closed-loop CC can't follow a scheduled capacity drop



Congestion window evolution with profiling-only:

- Window too slow to grow
- Can only shrink due to loss

Profiling can NOT save an inadequate CCA

Congestion induced head-of-line blocking:

store full
arrival cannot be admitted

→

gap in stream
in-order delivery stalls the app

→

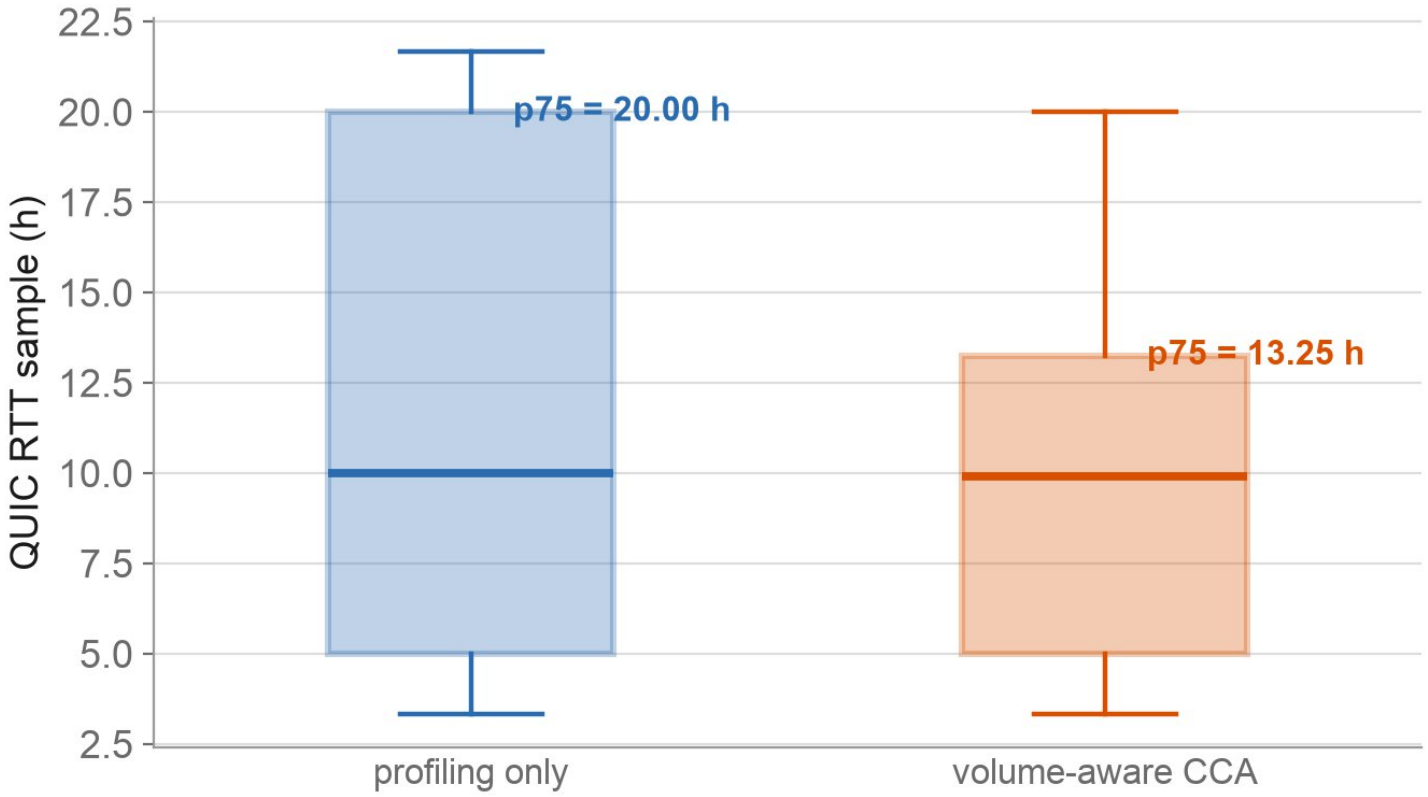
retransmit
queued behind the whole backlog

→

recovery
costs several contact windows

i2cat

Result 2: latency, and what it does not show

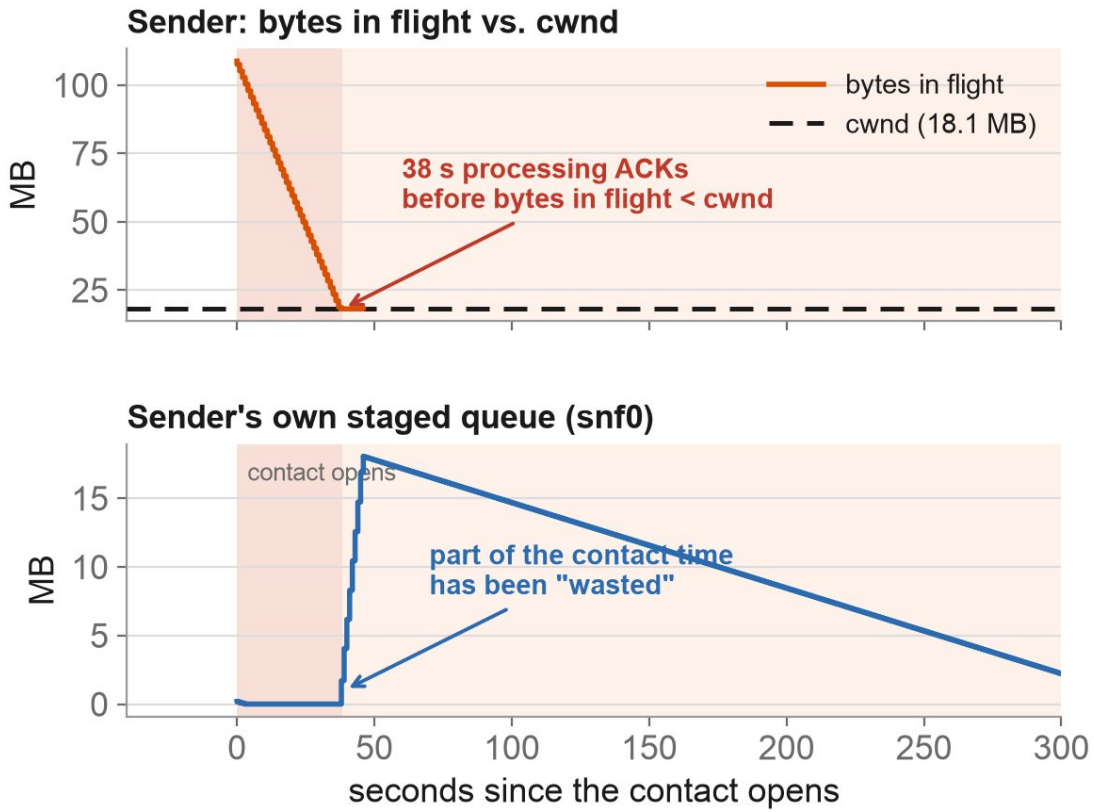


RTT samples as a **proxy** for delivery delay

Medians are identical (~10 h) — RTT here is dominated by the contact schedule, not by the CCA

PTO fires repeatedly — but harmlessly: it only sends a probe, never touching loss detection or the CCA

Why only 93.94 %?: the ACK-clearing delay



When zooming in to some contacts, their **initial seconds are spent unused**, with no packets awaiting forwarding.

Cause: the congestion window bounds **unacknowledged bytes** — not bytes still in the network.

→ **~2.5 %** difference w.r.t. UDP is explained by **QUIC overhead — the rest is ACK-clearing** time at the start of contacts

Limitations

The result holds in a **bounded regime**:

Oracle assumption — perfect a-priori knowledge.

Single flow — Model is extendable to multiple users, but remains untested

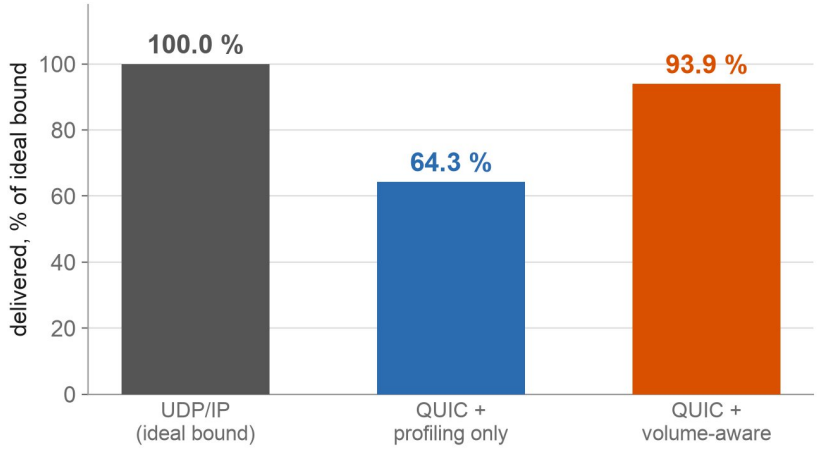
No Loss — Channels where assumed lossless at IP level

No multi-path — Single data mule

Interesting paths for future work

Takeaways

- **A profiled QUIC does not break** when the continuous end-to-end assumption is violated
- **Classic CCAs are unsavable** in challenged networks
- Sizing the window to **volume** maximizes delivered data with minimum congestion
- Congestion window must track **unacknowledged bytes, not network capacity**
- Profiling & volume-aware CCA: **RFC compliant, fully interoperable**
- **CGR as an inspiration**, not a directly applicable solution



i2cat

Research & Innovation Centre

Thank you!

oriol.fuste@i2cat.net

ns-3 QUIC: https://github.com/ENTEL-WNG/quic-ns-3

This work has been supported by Spanish MICIU/AEI/10.13039/501100011033 through projects PID2025-172002OA-I00 and PID2023-146378NB-I00; by Generalitat de Catalunya and the European Social Fund (Joan Oró Grants to hire research staff in training FI 2026, no. 2026 FI-3 00701); and by Government of Catalonia in the scope of the Space Strategy for Catalonia 2030.

Plaça de Pau Vila, 1
Palau de Mar Building, Sector B - 1st Floor
08039 Barcelona
Tel. (+34) 935 532 510

X/Twitter | Linkedin | YouTube | Bluesky | Instagram

Full contact plan

#	Link 0→1 feeding it	Link 1→2 contact	Path volume
1	0.17–0.25 h (300 s)	1.67–1.75 h	**18.75 MB**
2	3.33–3.42 h (300 s)	10.00–10.08 h	**18.75 MB**
3	13.33 / 16.67 / 20.00 h (3 × 1200 s)	22.83–23.00 h	**150 MB**
4	23.33 / 25.00 / 26.67 h (3 × 1200 s)	27.83–28.00 h	**150 MB**
5	28.33–28.58 h (900 s)	30.00–30.08 h	**56.25 MB**
6	31.67–31.75 h (300 s)	40.00–40.08 h	**18.75 MB**
7	45.00–45.50 h (1800 s)	46.67–47.17 h	**112.5 MB**
8	50.00–50.08 h (300 s)	53.33–53.50 h	**18.75 MB**

Σ = **543.75 MB** raw. UDP delivers 538.87 MB = 99.1 % of it.

Full profiled parameter set

Parameter	Value	Rationale
`IdleTimeout`	20 h	> longest scheduled disruption
`kInitialRtt`	4 h	~1.5× expected handshake RTT
`kInitialWindow`	12 500 pkts	= first path volume (18.125 MB)
Flow-control limits	≈ ∞	delegated entirely to the CCA
Buffers	2 × max volume	max deliverable volume, not BDP
0-RTT	pre-shared keys	no handshake RTT on reconnect
`max_ack_delay`, thresholds	spec defaults	left unchanged — stated for completeness

PTO behaviour — why it is benign

PTO fires repeatedly in both QUIC stacks — RTT swings far exceed what `smoothed_rtt` / `rtt_var` can predict

Profiling only: 3 firings (6.67 h, 35.01 h, 41.69 h), max count 2

Volume-sized cwnd: 2 firings (6.67 h, 35.00 h), max count 1

Why it is harmless: a PTO sends a *probe*. In QUIC it is decoupled from loss detection and from the congestion controller — it does not shrink cwnd and does not declare loss.

Contrast with TCP, where an RTO both retransmits and collapses the window. This is one of the concrete places where QUIC's design is better suited to this regime than TCP's.