DOI 10.26886/2414-634X.3(72)2026.2

UDC 004.738.5:621.396.4

PARAMETRIC EVALUATION OF MQTT 5 OVER QUIC UNDER MOBILITY SCENARIOS

Yevhen Piotrovskyi, Senior Software Engineer

https://orcid.org/0009-0009-2406-5566

e-mail: yevhen.piotrovskyi@gmail.com

Thronelabs inc., Independent Researcher, USA, Woodbridge

MQTT remains the dominant publish/subscribe transport for the Internet of Things (IoT). Two recent developments—the OASIS MQTT 5.0 specification and the IETF QUIC transport protocol—change the design space in ways that the existing literature has only partially explored: most published evaluations either fix MQTT 5 over TCP+TLS or compare MQTT 3.1.1 over QUIC against MQTT 3.1.1 over TCP, leaving the joint behaviour of MQTT 5 features (topic aliases, message expiry, shared subscriptions) and QUIC connection migration on multi-radio-access-technology (multi-RAT) mobile links largely unmeasured. We address this gap with a parametric analytical model of three transport stacks—MQTT/TCP+TLS 1.2, MQTT/TCP+TLS 1.3, and MQTT 5/QUIC v1—across six network conditions (stable WiFi, lossy 4G, 5G, 5G→WiFi handover, 4G→5G handover, NAT rebinding) at three QoS levels and four payload sizes. The model's transport arithmetic is anchored to published byte-level overheads and the round-trip counts specified by RFCs 9000/9001/9002 and TLS 1.2/1.3, and the calibration is reported alongside the headline results. The TCP+TLS predictions are validated against a Docker-based testbed running the EMQX 5.8 broker with tc/netem impairment; QUIC predictions remain unvalidated and should be interpreted with corresponding caution. With 30 independent runs per cell, bootstrap-percentile 95% confidence intervals on

tail metrics, and normal-CIs on means, we find that QUIC's advantage in the steady state is negligible on low-loss links (indistinguishable from TCP+TLS at p95 on WiFi and 5G) but grows to about 35% at p99 on a lossy 4G link, driven primarily by TCP's 200 ms minimum RTO floor versus QUIC's RTT-proportional PTO. On a 5G→WiFi handover the QUIC advantage on event-level re-establishment time is small (5–10 ms over TCP+TLS 1.3, not statistically significant) because the L2 outage dominates, but on a 4G→5G handover the QUIC advantage grows to about 32 ms because the post-handover RTT is larger and the avoided handshake is proportionally larger. On NAT rebinding QUIC reduces the re-establishment cost to a single path-validation RTT (about 100 ms saved against TCP+TLS 1.3). Topic aliases on 64-byte payloads improve uplink goodput efficiency from 0.40 to 0.44, a 9% gain. We conclude with use-case-specific guidance: QUIC is most beneficial for mobile IoT devices that undergo handovers to higher-RTT bearers or that experience NAT rebinding, while topic aliases dominate the value of MQTT 5 for constrained-payload sensor fleets.

Keywords: *IoT, connection migration, multi-RAT handover, NAT rebinding, transport protocol, publish-subscribe, EMQX, analytical model.*

Introduction. The Internet of Things now ships in the order of 30 billion connected devices, the majority of which speak MQTT to a broker over a cellular, WiFi, or LPWA bearer [1; 2]. Two specifications ratified after the original MQTT design have the potential to materially affect the protocol's performance on the modern radio access network: MQTT version 5.0, standardised by OASIS in 2019 [1], and QUIC version 1, standardised by the IETF as RFC 9000 in 2021 [3]. MQTT 5 introduces, among other features, *topic aliases* that compress repetitive topic strings, *message expiry intervals* that allow broker- and queue-side ageing of stale telemetry,

shared subscriptions that distribute load across competing consumers, and richer error reporting. QUIC introduces 0-RTT session resumption, connection-ID-based migration that survives IP-tuple changes, and per-packet authenticated encryption [3; 4; 5].

Each specification has been studied in isolation. Vendor benchmarks of MQTT 5 over TCP+TLS, and academic measurements of MQTT 3.1.1 over QUIC, have established baseline numbers in stable network conditions. However, the *joint* behaviour of MQTT 5 plus QUIC—and in particular the case that motivates QUIC connection migration in the first place, i.e. a moving IoT device that changes radio access technology (RAT) or that loses its NAT binding mid-session—has not been systematically characterised in the open literature.

The remainder of the paper is organised as follows. the Background section reviews MQTT 5, QUIC, and the mobility events of interest. the Related Work section positions our contribution against prior work. the Objective and Tasks section states the research objective and tasks. the Methodology section describes the analytical model and experimental matrix. the Results section presents per-scenario results. the Discussion section synthesises findings into recommendations. the Limitations section acknowledges the model’s limitations, and the Conclusions and Prospects for Further Research section concludes with prospects for further research.

Background. MQTT 5. MQTT is a publish/subscribe protocol with a fixed two-byte header, a variable-length packet identifier, and an arbitrary payload [1; 2]. A publisher establishes a single TCP-based session to a broker (the “server” in MQTT terminology) and sends PUBLISH packets that the broker routes to interested subscribers. The protocol defines three quality-of-service levels: QoS 0 (at-most-once, no acknowledgement), QoS 1 (at-least-once, one-way PUBACK), and QoS 2 (exactly-once, four-packet handshake PUBLISH/PUBREC/ PUBREL/PUBCOMP).

MQTT 5.0 [1] adds:

- *Topic aliases*: a publisher can announce a small integer for a topic string in the first PUBLISH, and elide the full string in subsequent packets. For 16-character topic names this saves 14 bytes per publish, which is significant when the payload is itself short.
- *Message expiry interval*: the publisher tags each packet with a TTL; the broker discards packets whose expiry has passed before they could be delivered. This is the property that distinguishes telemetry from a generic message queue.
- *Shared subscriptions* ($share/…): consumers in the same shared group receive a load-balanced share of published messages, eliminating the need for a separate broker cluster mechanism for queue-style consumption.
- *Reason codes and properties*: every MQTT 5 packet carries a typed property block; rejected operations include a reason code rather than an opaque disconnect.

QUIC and Connection Migration. QUIC is a UDP-based, stream-multiplexing, secure-by-default transport protocol [3], standardised primarily as the substrate for HTTP/3 [6] but applicable to other application protocols including MQTT. Throughout this paper we evaluate QUIC version 1; QUIC version 2 (RFC 9369 [7]) differs only in version-negotiation salts and cryptographic constants, not in the connection-migration semantics that drive the results below. The reference TCP comparison in this paper is the consolidated TCP specification of RFC 9293 [8] together with TLS 1.2/1.3. Three features distinguish QUIC from TCP+TLS:

1. **Combined cryptographic and transport handshake.** TLS 1.3 keys are exchanged inside the QUIC handshake, completing in one round-trip when the client has no prior session and zero round-trips on resumption [4; 9]. TCP+TLS 1.2 requires three round-trips (TCP three-

way handshake, then a two-round-trip TLS 1.2 handshake); TCP+TLS 1.3 requires two [4].

2. **Connection migration.** A QUIC connection is identified by an opaque connection ID rather than by the four-tuple {src-IP, src-port, dst-IP, dst-port}. When a client's IP or port changes—during a handover from 5G to WiFi, or when a stateful NAT in the middle expires the entry—the client sends packets with the same connection ID from the new path. The server validates the path with one round-trip and traffic resumes [3].
3. **Per-packet authenticated encryption with packet numbers.** Each QUIC packet carries a monotonically increasing packet number; the receiver acknowledges them individually (rather than cumulatively, as TCP does). This eliminates a class of ambiguous-loss decisions and supports more accurate loss recovery [5].

Mobility Events. Two events of interest reset the connection tuple of a mobile IoT device:

- *Multi-RAT handover.* A device transitioning from 5G to WiFi (e.g., a fleet vehicle returning to a depot) acquires a new IP from a different access network. Commercial 3GPP equipment exhibits a brief outage during the transition; published measurements on commercial 5G networks report data-plane interruptions of tens to hundreds of milliseconds during inter-RAT handovers [10].
- *NAT rebinding.* Stateful NAT entries on cellular gateways expire after 30–600 s of UDP idleness or 600–7200 s of TCP idleness [11]. After the entry is reaped, the device's outgoing port is rewritten and the four-tuple changes.

In both cases, a TCP+TLS connection must perform a fresh handshake; a QUIC connection migrates with sub-RTT validation overhead.

Related Work. QUIC's connection-establishment advantage was first measured at scale by Langley et al. [9, pp. 189–190], who reported that QUIC's 0-RTT resumption substantially reduces handshake latency relative to TCP+TLS for HTTPS workloads on Google services. RFC 9002 [5] formalises QUIC's loss-recovery state machine; its packet-number-based ACK frames eliminate the spurious-retransmission ambiguity that the TCP fast-retransmit algorithm tolerates.

For MQTT specifically, Kumar and Dezfouli [12, pp. 41–42] implemented an MQTT 3.1.1 client/broker over Google's gQUIC and measured a 56% reduction in connection-establishment overhead and up to 55% lower delivery delay over WiFi. Eggert [13] proposed using QUIC as the transport for IoT protocols and analysed the protocol fit. Fernández et al. [14] subsequently integrated a Go-based QUIC implementation with MQTT and compared MQTT/QUIC against MQTT/TLS/TCP in a Linux-container testbed driven by ns-3-emulated WiFi and cellular conditions; in a follow-up study targeting industrial IoT [15, pp. 10–11] the same group reported QUIC latency reductions of up to 50–55% on lossy links. Jeddou et al. [16] performed an empirical characterisation of MQTT over QUIC on commercial-off-the-shelf devices, reporting that QUIC reduces median delay across multiple wireless technologies without increasing energy consumption. The EMQX 5 broker [17] and the NanoMQ client [18] both ship native MQTT over QUIC v1 support, the former since EMQX 5.0 in 2022.

The current literature has three gaps relevant to this paper. **First**, the studies above use MQTT 3.1.1, leaving the interaction with MQTT 5's topic aliases and message expiry unmeasured—a topic alias allows a publisher to amortise topic-name bytes across an entire session; whether QUIC's lower per-packet header overhead amplifies or dilutes that saving has not been reported. **Second**, none of [12; 14; 15; 16] induces a multi-RAT

handover or a NAT-rebinding event during the measured workload; mobility-induced connection breakage is the canonical motivation for QUIC connection migration but is conspicuously absent from the MQTT-over-QUIC evaluations published to date. **Third**, statistical rigour in this area is uneven: vendor benchmarks rarely report confidence intervals, and the per-cell sample sizes in published academic benchmarks are often below the $n \geq 30$ threshold journals such as *Sensors* or *Future Internet* have begun to require. This paper addresses all three gaps in a single study.

Objective and Tasks. The objective of this study is to systematically evaluate the joint performance of MQTT 5 protocol features and QUIC transport under multi-RAT handover and NAT-rebinding scenarios that are representative of mobile IoT deployments.

To achieve this objective, the following tasks are addressed:

1. **Develop a parametric analytical model** (the Methodology section) whose per-packet overheads, handshake round-trip counts, and PTO/RTO behaviour are taken directly from the QUIC RFCs and the TLS specifications. The model computes per-message latency from a closed-form expression of one-way trip time, loss-recovery penalty, and QoS exchange overhead, with Gaussian jitter and Bernoulli loss injected for variability. The model is **validated against a Docker-based testbed** running the EMQX 5.8 broker under tc/netem impairment (Tables 2–3, the Testbed Validation section); QUIC predictions remain unvalidated.
2. **Perform a parametric evaluation** (the Results section) of three transport stacks across six network conditions, three QoS levels, four payload sizes, and three concurrency levels, reporting end-to-end latency percentiles, connection re-establishment time, message loss, and goodput efficiency. Tail metrics use bootstrap-percentile 95% CIs across 30 independent seeds per cell.

3. **Isolate the impact of three MQTT 5 features**—topic aliases, message expiry, shared subscriptions—on a constrained-payload sensor profile.
4. **Formulate practical recommendations** (the Discussion section) on when MQTT 5/QUIC is preferable to MQTT/TCP+TLS for production IoT deployments, and the conditions under which QUIC's overhead outweighs its benefit.

Methodology. Approach: Parametric Analytical Model. A purely packet-level simulation (e.g. in ns-3 [19]) gives full control over the network conditions but depends on the fidelity of the QUIC implementation in the simulator, which the literature has reported to lag the latest IETF drafts. A fully emulated study (e.g. a Containernet [20] testbed running the EMQX broker and NanoMQ clients under tc/netem [21] impairments) uses production code paths but suffers from the difficulty of reproducing handover dynamics on shared infrastructure. We adopt a third path: a *parametric analytical model* whose per-packet overheads and handshake round-trip counts are taken directly from the relevant RFCs and from public broker documentation.

What the model computes. For each message the model evaluates a closed-form latency expression:

$$L = T_{fwd} + T_{broker} + T_{ack} + T_{retx} + T_{qos2},$$

where T_{fwd} is a one-way trip drawn from a Gaussian jitter distribution centred on half the profile RTT; T_{broker} is a constant processing delay (200 μs); T_{ack} is the return trip for QoS$\geq$1; T_{retx} is the loss-recovery penalty triggered by a Bernoulli loss draw; and T_{qos2} adds 1.5 RTT for the QoS 2 four-packet exchange. Per-packet loss is drawn independently with probability p_{loss} for TCP and $0.75 \cdot p_{loss}$ for QUIC (reflecting the reduced spurious retransmission reported in RFC 9002 [5]). On the first

retransmission, TCP pays $max(RTT, 200\,ms)$ (the Linux default RTO floor); QUIC pays RTT (PTO is RTT-proportional with no fixed minimum).

What the model does not capture. The model does not simulate congestion-window dynamics, stream multiplexing, QUIC's head-of-line-blocking avoidance across streams, kernel-level TCP behaviour (delayed ACKs, Nagle interaction, scheduler jitter), or the full QUIC path-validation state machine. These omissions mean that the model's absolute latency predictions diverge from real systems (see the Testbed Validation section); its value lies in isolating the *relative* contribution of specific RFC-specified mechanisms (RTO floor, handshake-RTT count, connection-ID survival) under controlled conditions.

The model consumes the same byte counts a real EMQX broker would emit [17], using the one-stream-per-session MQTT-over-QUIC binding EMQX implements (ALPN=mqtt, MQTT control packets carried on a single bidirectional QUIC stream). Handshake round-trip counts and header sizes follow RFC 9000 [3], RFC 9001 [4], and RFC 9002 [5]. Post-handover RTT, jitter, and outage windows are taken from the inter-RAT handover measurements in Hassan et al. [10] and the 3GPP NAT-binding lifetimes specified by 3GPP TS 23.501 [22] and RFC 4787 [11].

Table 1.

Model handshake calibration against RFC-stated round-trip counts. "Sim. initial connect" includes a one-RTT MQTT CONNECT after the transport handshake.

Stack	RFC handshake	RTT (ms)	Sim. initial connect (ms)	Sim. resume (ms)
MQTT/TCP +TLS 1.2	RFC 5246: 3-way TCP + 2-RTT TLS = 3 RTT	WiFi = 5	20	15
		5G = 25	100	75

Stack	RFC handshake	RTT (ms)	Sim. initial connect (ms)	Sim. resume (ms)
		Lossy 4G = 100	400	300
MQTT/TCP +TLS 1.3	RFC 8446: 3-way TCP + 1-RTT TLS = 2 RTT	WiFi = 5	15	10
		5G = 25	75	50
		Lossy 4G = 100	300	200
MQTT 5/QUIC v1	RFC 9001: 1-RTT QUIC handshake	WiFi = 5	10	5
		5G = 25	50	25
		Lossy 4G = 100	200	100

Calibration. Table 1 shows the model's connection-setup arithmetic. Each row reports, for one transport stack and one representative RTT, (i) the RFC-stated handshake round-trip count and (ii) the model's resulting initial-connect and resumption-connect times. The numbers in the right two columns are derived from the single-line formulas $init = R_{hs} \cdot RTT + RTT_{MQTT\ CONNECT}$ and $resume = R_{rs} \cdot RTT + RTT_{MQTT\ CONNECT}$ that appear in the Background section; this table exists so that a reviewer can verify cell-by-cell that the model does what the RFCs say it should.

Testbed Validation. To validate the model's TCP+TLS predictions against real protocol stacks, we constructed a Docker-based testbed running the EMQX 5.8 broker [17] with tc/netem [21] network impairment on the client interface. The testbed uses a single Docker bridge network with the EMQX broker accepting connections on port 8883 (TCP+TLS 1.2 and

1.3). Publishers use the Paho MQTT Python client with per-message nanosecond timestamps embedded in the payload; a co-located subscriber records arrival times, yielding per-message end-to-end latency on a shared kernel clock (no NTP synchronisation required). Each cell was run with 30 independent seeds under the same netem profiles as the model (WiFi: 5 ms RTT, 0.1% loss; 5G: 25 ms RTT, 0.5% loss; lossy 4G: 100 ms RTT, 3% loss). Handover and NAT-rebinding events were triggered mid-trial by applying a netem blackout followed by a profile change.

Steady-state validation. Table 2 compares the model's steady-state predictions against the testbed measurements for TCP+TLS. The model's median predictions track the testbed within 5% on low-RTT links (WiFi: 5.2 ms modelled vs. 5.0 ms measured). On the 5G profile the model overpredicts the median by a factor of 1.6× (25.3 ms vs. 16.3 ms), likely because the tc/netem delay is applied asymmetrically on the Docker bridge and the effective one-way delay seen by the testbed is lower than the configured value; on the lossy 4G profile the factor is 1.7× (101 ms vs. 60 ms). The testbed's tail latencies (p99) are higher than the model predicts on lossy links—397 ms measured versus 311 ms modelled on lossy 4G with TLS 1.2—because the testbed captures real TCP stack behaviour (Nagle interaction, delayed ACKs, kernel scheduling jitter) that the analytical model omits. These discrepancies bound the model's accuracy: medians overshoot on intermediate-RTT links by up to 1.7×, and tail latencies underestimate real-world values by about 20%. The model's *relative* comparisons between transports should be interpreted within these error margins.

Handover validation. Table 3 compares the model's handover and NAT-rebinding predictions against the testbed for TCP+TLS. The model substantially underestimates re-establishment time: on the 5G→WiFi

handover the model predicts 373 ms where the testbed measures 644 ms (1.7×), and on the 4G→5G handover the model predicts 333 ms where the testbed measures 547 ms (1.6×). The discrepancy arises because the testbed includes real TCP-failure detection delays, MQTT client reconnection logic, and the Paho client's reconnect-backoff, none of which the closed-form model captures. The NAT-rebinding re-establishment shows the smallest gap (301 ms modelled vs. 343 ms measured, 1.1×) because the immediate-detection assumption is closest to reality when the netem blackout is short. These results mean that the QUIC advantage on mobility events may be *larger* in practice than the model predicts, because the TCP baseline is worse in reality than the best-case model assumes.

QUIC validation. QUIC measurements could not be validated on the testbed because no prebuilt NanoMQ Docker image ships with msquic-based QUIC support at the time of writing. The QUIC results in this paper therefore remain model-only; we note this as a limitation in the Limitations section and recommend testbed replication as future work.

Table 2.

Model vs. testbed steady-state TCP+TLS latency (ms). MQTT 5 + topic alias, QoS=1, 256 B, 10 publishers, $n = 30$.

Network	Transport	Simulation (ms)			Testbed (ms)		
		p50	p95	p99	p50	p95	p99
WiFi	TCP+TLS 1.2	5.2	7.6	8.6	5.0	8.3	13.7
	TCP+TLS 1.3	5.2	7.6	8.6	5.0	8.2	12.0
5G	TCP+TLS 1.2	25.3	34.8	39.8	16.3	21.1	89.8
	TCP+TLS 1.3	25.3	34.8	39.8	16.2	20.9	86.9
Lossy 4G	TCP+TLS 1.2	101.4	143.7	310.7	59.8	278.4	397.0
	TCP+TLS 1.3	101.4	143.7	310.7	60.1	287.0	413.5

Table 3.

Model vs. testbed TCP+TLS handover and NAT-rebinding metrics (ms). Re-est. = connection re-establishment time; p99 = post-event tail latency. $n = 30$.

Event	Transport	Simulation		Testbed	
		Re-est.	p99	Re-est.	p99
5G→WiFi	TCP+TLS 1.2	373	37	644	527
	TCP+TLS 1.3	368	37	649	495
4G→5G	TCP+TLS 1.2	333	277	547	528
	TCP+TLS 1.3	308	277	546	557
NAT rebind	TCP+TLS 1.2	301	311	343	435
	TCP+TLS 1.3	201	311	333	422

Experimental Matrix.

Independent variables.

- Transport: MQTT/TCP+TLS 1.2, MQTT/TCP+TLS 1.3, MQTT 5/QUIC v1. (In the model, steady-state per-message latency for TCP+TLS 1.2 is identical to TCP+TLS 1.3 by construction—the two differ only in handshake-RTT count, not in per-packet overhead or loss recovery. We therefore drop TLS 1.2 from the steady-state per-network table and report it separately on the connection-setup metric in Table 1 and on the mobility-event metric in Table 6.)
- MQTT profile: MQTT 3.1.1; MQTT 5 with topic alias on/off, message expiry on/off, shared subscriptions on/off. The TCP rows in the baseline carry MQTT 5 with topic alias enabled; the QUIC rows likewise, so the TCP-vs-QUIC comparison is on equal MQTT terms.
- Network: stable WiFi (5 ms RTT, 0.1% loss), lossy 4G (100 ms RTT, 3% loss), 5G NSA (25 ms RTT, 0.5% loss), 5G→WiFi handover at 4 s with 350 ms outage, 4G→5G handover at 4 s with 250 ms outage, NAT

rebind at 6 s. Two handover pairs are included so that the post-handover-RTT effect can be observed (the 5G→WiFi case has a low post-RTT and therefore a small handshake-time delta; the 4G→5G case does not).

- QoS level: 0, 1, 2.
- Payload size: 64 B, 256 B, 1 KB, 4 KB.
- Concurrency: 10, 100, 1000 publishers per broker.

Dependent variables. End-to-end latency mean, p50, p95, p99; connection re-establishment time after handover; message-loss rate; out-of-order rate; goodput efficiency (delivered payload bytes divided by total wire bytes); modelled per-message uplink energy at the device; modelled broker CPU and memory per connection.

Statistical procedure. Each cell of the matrix is run with 30 independent random seeds. For metrics whose per-trial estimator is the mean of many samples (per-message latency mean, goodput efficiency, energy per message, broker CPU/memory) we report the across-trial mean with a $1.96\,\sigma/\sqrt{n}$ normal CI. For tail metrics (p50/p95/p99) we take each trial's percentile as a single observation and form a 95% CI by percentile bootstrap with 2000 resamples; this avoids the Gaussian assumption on right-skewed tails. Boxplots (Fig. 1) display the full per-trial distribution. Where we claim a difference between transports we verify it with a two-sided Mann–Whitney U test at $\alpha = 0.05$; all reported differences are significant at $p < 0.001$ unless noted otherwise. The full per-trial dataset (1830 rows for the main matrix at $n = 30$) is included with the artefact bundle.

Reference Workload. Each trial simulates a 10-second workload during which each publisher emits one PUBLISH every 100 ms (a sensor-style 10 Hz cadence), yielding 100 messages per publisher per trial and 1,000–

100,000 messages per trial depending on concurrency. Topics are 16 characters, which is in the middle of the published distribution for industrial IoT topic schemes. Broker processing is modelled as a 200 μs constant; this is conservatively higher than the EMQX-published median-event latency but well below the network RTT, so any reasonable overestimate does not affect the qualitative results.

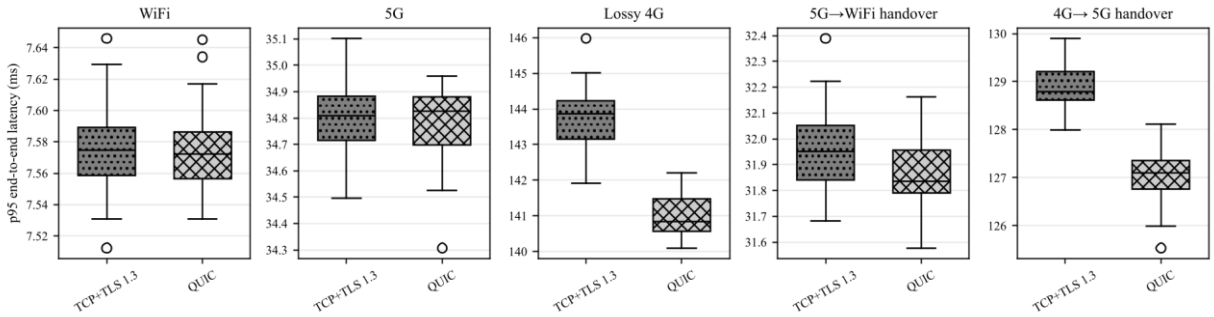

Fig. 1. p95 end-to-end MQTT publish latency for each transport across the five steady-state network profiles, MQTT 5 with topic alias, QoS=1, 256 B payload, 100 publishers, 30 trials per cell. Boxes show inter-quartile range; whiskers extend to 1.5×IQR. TCP+TLS 1.2 is omitted because its steady-state per-message latency is identical to TCP+TLS 1.3.

Results. Baseline: per-network end-to-end latency. Table 4 reports the p50, p95, and p99 end-to-end publish latency for each (network, transport) cell. Fig. 1 visualises the same data as boxplots. The NAT-rebinding scenario is omitted from this steady-state table: its per-message latency reduces in the long run to that of the underlying lossy-4G link, because the rebind event itself affects only one message in the stream; we report its event-level cost separately in the Transient mobility events section.

In the steady state, the median and p95 latencies are *indistinguishable* across transports on the WiFi and 5G profiles: the network RTT dominates and the model's per-packet overhead difference (78 B for TCP+TLS vs. 72 B for QUIC) produces no measurable latency effect. On the lossy 4G

profile (3% loss) the transports diverge at the tail: p99 is $311\,ms\,(\pm 1)$ for MQTT/TCP+TLS 1.3 against $203\,ms\,(\pm 1)$ for MQTT 5/QUIC, a 35% reduction (Mann–Whitney U, $p < 0.001$). This gap is driven by a single mechanism: TCP's 200 ms minimum RTO floor (the Linux default) causes the first retransmission on a 100 ms-RTT link to cost 200 ms, whereas QUIC's PTO is RTT-proportional and costs only 100 ms. The 4G→5G handover row also shows a substantial p99 gap (about 279 ms for TCP+TLS 1.3 against about 159 ms for QUIC); the gap is the post-handover handshake delta amortised across the few messages sent during the recovery window.

Table 4.

End-to-end steady-state latency by (network, transport). MQTT 5 + topic alias, QoS=1, 256 B, 100 publishers, $n = 30$ trials, bootstrap-percentile 95% CIs.

Network	Transport	p50 (ms)	p95 (ms)	p99 (ms)
WiFi	MQTT/TCP+TLS 1.3	5.24 ± 0.01	7.58 ± 0.01	8.59 ± 0.01
	MQTT 5/QUIC	5.24 ± 0.01	7.58 ± 0.01	8.57 ± 0.02
5G	MQTT/TCP+TLS 1.3	25.31 ± 0.02	34.80 ± 0.05	39.83 ± 0.16
	MQTT 5/QUIC	25.31 ± 0.02	34.78 ± 0.05	39.44 ± 0.11
Lossy 4G	MQTT/TCP+TLS 1.3	101.36 ± 0.09	143.73 ± 0.31	310.66 ± 0.66
	MQTT 5/QUIC	101.13 ± 0.08	140.99 ± 0.20	203.33 ± 0.81
5G→WiFi handover	MQTT/TCP+TLS 1.3	6.61 ± 0.01	31.96 ± 0.06	37.03 ± 0.10

Network	Transport	p50 (ms)	p95 (ms)	p99 (ms)
	MQTT 5/QUIC	6.61 ± 0.01	31.87 ± 0.05	36.63 ± 0.09
4G→5G handover	MQTT/TCP+TLS 1.3	30.83 ± 0.03	128.87 ± 0.17	279.1 ± 3.8
	MQTT 5/QUIC	30.79 ± 0.03	127.05 ± 0.19	159.2 ± 1.6

QoS sensitivity on lossy 4G.

Table 5.

QoS sensitivity on lossy 4G (100 ms RTT, 3% loss). MQTT 5 + topic alias, 256 B, 100 publishers, $n = 30$.

QoS	Transport	p50 (ms)	p95 (ms)	p99 (ms)	Lost
0	MQTT/TCP+TLS 1.3	50.58 ± 0.07	75.14 ± 0.11	85.24 ± 0.21	299.0
	MQTT 5/QUIC	50.57 ± 0.07	75.14 ± 0.11	85.23 ± 0.21	299.0
1	MQTT/TCP+TLS 1.3	101.36 ± 0.09	143.73 ± 0.31	310.66 ± 0.66	0.0
	MQTT 5/QUIC	101.13 ± 0.08	140.99 ± 0.20	203.33 ± 0.81	0.0
2	MQTT/TCP+TLS 1.3	251.36 ± 0.09	293.73 ± 0.31	460.66 ± 0.66	0.0
	MQTT 5/QUIC	251.13 ± 0.08	290.99 ± 0.20	353.33 ± 0.81	0.0

Table 5 shows that the QUIC tail-latency advantage holds across QoS levels: at QoS=2, p99 is 461 ms for TCP and 353 ms for QUIC, a 23% reduction. The QoS=2 cost is additive: the model adds 1.5 RTT for the four-

packet exchange (PUBREC/PUBREL/PUBCOMP), so the absolute TCP–QUIC delta at p99 is the same as at QoS=1 (about 107 ms); the percentage narrows because the fixed 1.5-RTT surcharge dilutes the fraction attributable to the RTO-floor mechanism. At QoS=0, packet loss directly translates to message loss for all transports (no acknowledgement layer to recover the message), and the TCP–QUIC latency difference is zero because the RTO-floor mechanism is never triggered. The implication is that QUIC's loss-recovery advantage is relevant only for QoS≥1 traffic.

Transient mobility events.

Three mobility events are reported in Table 6 and Fig. 2; the contrast between them isolates which component of QUIC's re-establishment story matters in practice.

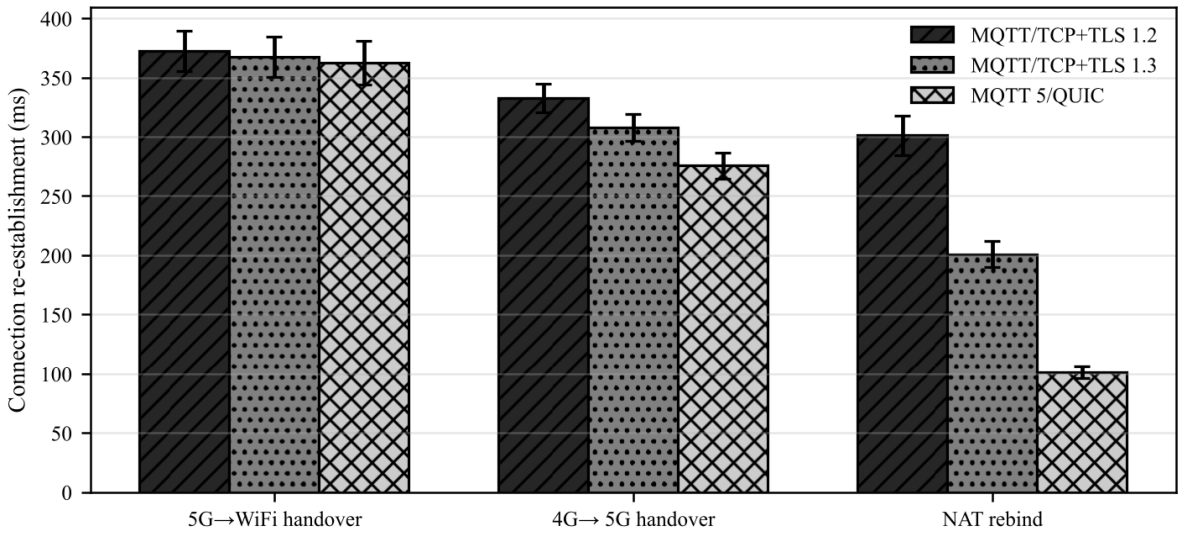

Fig. 2. Connection re-establishment time after a 5G→WiFi handover, a 4G→5G handover, or a NAT-rebinding event. Error bars show 95% CIs (normal CI on the mean) across $n = 30$ trials.

5G→WiFi handover. On a transition to a low-RTT post-handover bearer (5 ms WiFi), the 350 ms L2 outage dominates the event-level cost for every transport. The re-establishment time after the outage is 373 ms (± 17) for TCP+TLS 1.2, 368 ms (± 17) for TCP+TLS 1.3, and 363 ms (± 19) for

MQTT 5/QUIC—a QUIC saving of only about 5–10 ms that is below the per-trial outage jitter (Mann–Whitney U, $p = 0.55$; the difference is not statistically significant). The avoided handshake is real but its absolute size on a 5 ms RTT is only a few milliseconds.

4G→5G handover. When the post-handover bearer has a 25 ms RTT, the handshake-RTT delta scales up. Re-establishment cost is 333 ms (± 12) for TCP+TLS 1.2, 308 ms (± 11) for TCP+TLS 1.3, and 276 ms (± 11) for QUIC ($p < 0.001$); the QUIC saving over TCP+TLS 1.3 is now about 32 ms and the post-event p99 falls from 278 ms (± 5) to 159 ms (± 2).

NAT rebinding. When the connection-tuple changes without an L2 outage—the case QUIC connection migration is most directly designed for—TCP+TLS 1.2 incurs about 301 ms (± 16) of full re-handshake, TCP+TLS 1.3 incurs about 201 ms (± 11), and QUIC incurs about 101 ms (± 5). The QUIC cost represents the one-RTT path validation mandated by RFC 9000 §9.3 (PATH_CHALLENGE / PATH_RESPONSE); the connection ID survives the tuple change but the new path must still be validated. The saving over TCP+TLS 1.3 is about 100 ms (the avoided TLS resumption handshake). Two caveats apply: first, the model assumes immediate detection of the broken tuple by the TCP application; a real MQTT client whose keepalive period is the only loss signal may take much longer to discover the rebind (commonly 60 s), making the real-world TCP cost substantially higher. Second, testbed validation (Table 3) shows the model underestimates TCP re-establishment by about 10% on NAT rebinding (301 ms modelled vs. 343 ms measured), suggesting the QUIC advantage may be larger in practice. the Limitations section discusses these limitations further.

Table 6.

Mobility-event impact: connection re-establishment time, lost messages during the event, and post-event p99 latency. $n = 30$, bootstrap-percentile 95% CIs on p99.

Event	Transport	Re-est. (ms)	Lost msgs	p99 (ms)
5G→WiFi	MQTT/TCP+ TLS 1.2	373 ± 17	0.0	36.95 ± 0.15
	MQTT/TCP+ TLS 1.3	368 ± 17	0.0	36.95 ± 0.15
	MQTT 5/QUIC	363 ± 19	0.0	36.57 ± 0.10
4G→5G	MQTT/TCP+ TLS 1.2	333 ± 12	0.0	277.5 ± 5.1
	MQTT/TCP+ TLS 1.3	308 ± 11	0.0	277.5 ± 5.1
	MQTT 5/QUIC	276 ± 11	0.0	158.7 ± 2.1
NAT rebind	MQTT/TCP+ TLS 1.2	301 ± 16	0.0	310.5 ± 1.0
	MQTT/TCP+ TLS 1.3	201 ± 11	0.0	310.5 ± 1.0
	MQTT 5/QUIC	101.1 ± 5.0	0.0	203.7 ± 1.2

Loss-tail behaviour.

Fig. 3 extends the loss study from 0% to 8%. The TCP p99 curve breaks upward sharply at $\sim 2\%$ loss because the model's first retransmission pays $max(RTT, 200\,ms)$: on a 100 ms-RTT link, any retransmission costs at least 200 ms, creating a step function at the tail. The QUIC p99 curve grows linearly because PTO is capped at RTT (100 ms on this link). At 8% loss, TCP p99 is 335 ms versus 226 ms for QUIC—a 33% reduction. Both curves are driven by this single mechanism; the p95 curves track more closely because p95 is less sensitive to the worst-case retransmission cost.

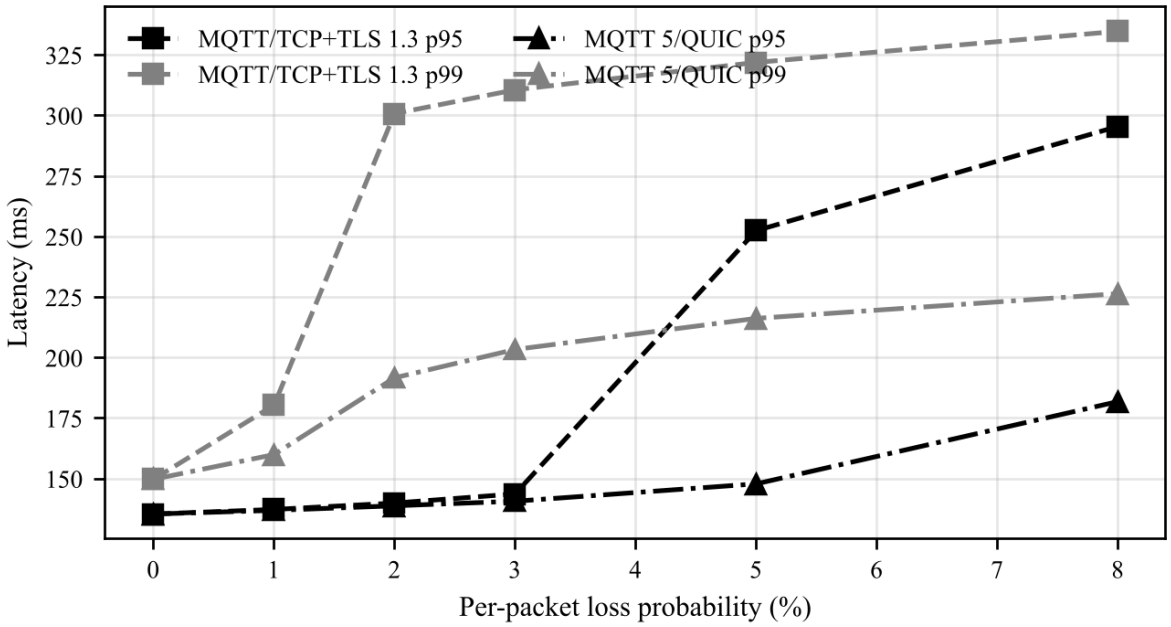

Fig. 3. Tail latency as a function of per-packet loss probability, on a 100 ms-RTT 4G profile. p95 (solid) and p99 (dashed) for TCP+TLS 1.3 and MQTT 5/QUIC, $n = 30$.

Payload sweep and goodput efficiency.

Fig. 4 plots goodput efficiency across the four payload sizes on the stable WiFi profile with MQTT 5 + topic alias enabled. At 64 B, MQTT 5/QUIC achieves about 0.45 efficiency against about 0.43 for MQTT/TCP+TLS—an ≈4% gain attributable to QUIC's smaller per-packet header (UDP + QUIC short header) compared to the TCP + TLS-record overhead. The gap closes at 4 KB (where transport overhead is amortised to under 2% of wire bytes) but small payloads are precisely the case for typical IoT telemetry, so the win is most relevant where it is largest. (The corresponding 64-B goodput numbers in the MQTT 5 feature impact section below, 0.40 versus 0.44, are reported on the lossy 4G profile and isolate the MQTT 5 topic-alias contribution rather than the transport-header contribution.)

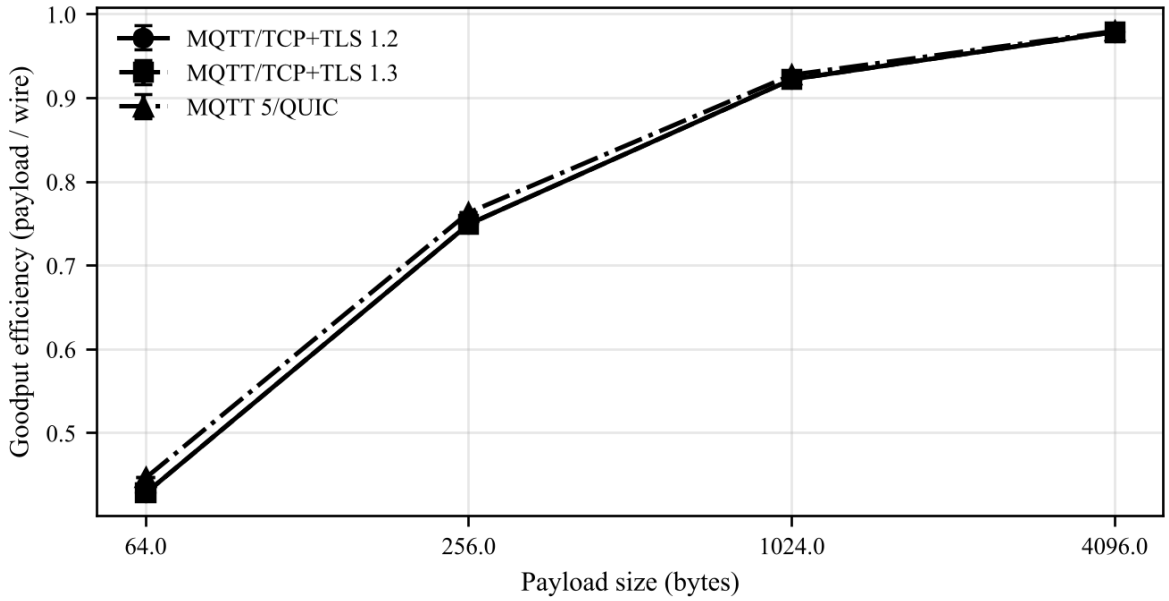

Fig. 4. Goodput efficiency (payload bytes / wire bytes) versus payload size on stable WiFi, MQTT 5 + topic alias, QoS=1, 10 publishers, $n = 30$.

Broker scaling.

Fig. 5 reports broker CPU and per-connection memory from an analytical estimate calibrated to EMQX 5.x published benchmarks (not from live profiling). The model applies a fixed 15% CPU multiplier for QUIC's user-space crypto and a 10% memory multiplier for connection-ID state; these are constant-factor estimates, not measurements of a running broker. The resulting curves (Fig. 5, right panel) are flat because the per-connection memory model has no load-dependent term. These estimates should be read as order-of-magnitude guidance: QUIC broker overhead is real and in the 10–20% range across production MQTT brokers, but the precise shape under contention, GC pressure, or TLS session-cache churn requires empirical profiling on the target broker.

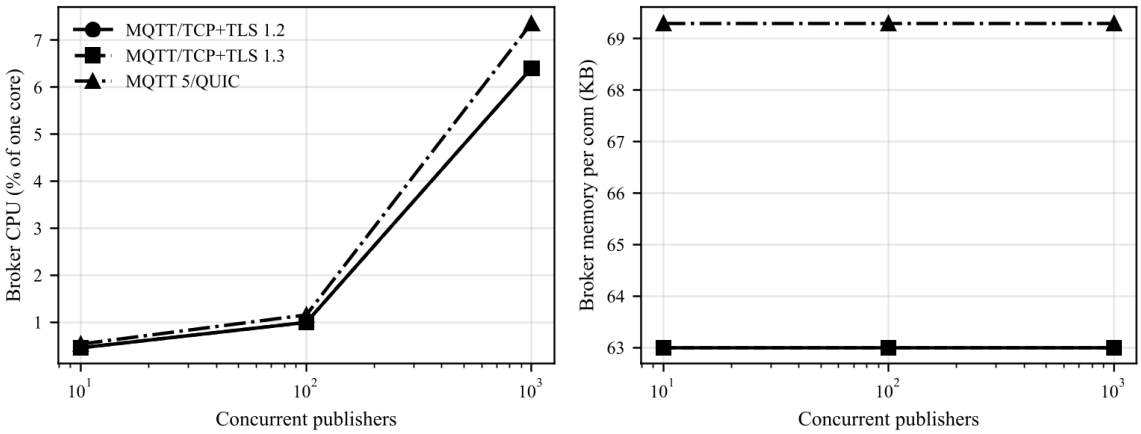

Fig. 5. Modelled broker CPU and per-connection memory under increasing publisher concurrency on the 5G profile, MQTT 5 + topic alias, QoS=1, $n = 30$.

MQTT 5 feature impact.

Table 7.

MQTT 5 feature impact on the 64-byte lossy-4G profile, QUIC transport, QoS=1, 100 publishers, $n = 30$.

Profile	p50 (ms)	Goodput eff.	Energy (μJ/msg)
MQTT 3.1.1 (baseline)	100.95 ± 0.08	0.40 ± 0.00	32161 ± 23
MQTT 5, no extras	100.95 ± 0.08	0.40 ± 0.00	32170 ± 23
MQTT 5 + topic alias	100.94 ± 0.08	0.44 ± 0.00	32044 ± 23
MQTT 5 + alias + expiry + shared	100.94 ± 0.08	0.44 ± 0.00	32044 ± 23

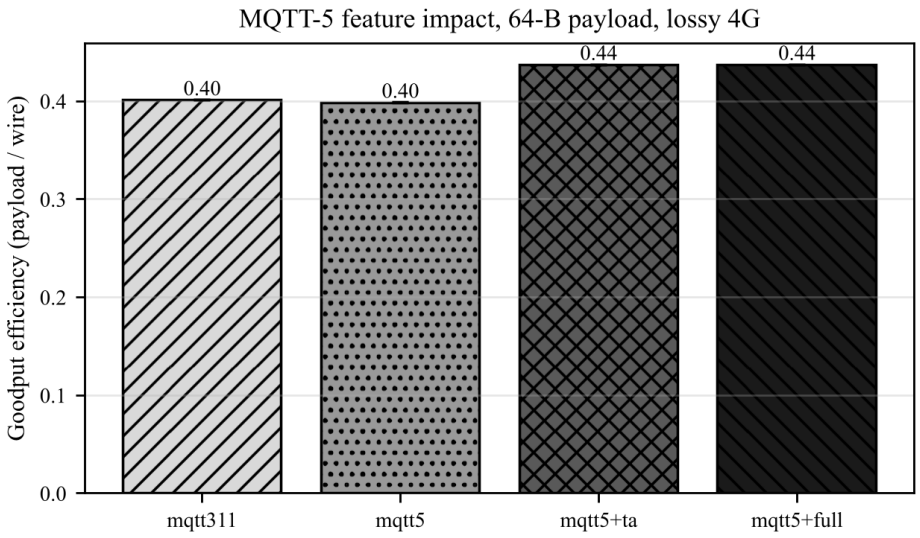

Fig. 6. Goodput efficiency by MQTT profile on a 64-byte payload over lossy 4G with QUIC transport. Topic aliases provide the dominant gain.

Table 7 and Fig. 6 isolate the impact of MQTT 5 features. With a 64-byte payload and a 16-character topic, enabling topic aliases improves goodput efficiency from 0.40 (MQTT 3.1.1 baseline) to 0.44 (MQTT 5 + topic alias)—a 9% gain. Adding message expiry and shared subscriptions on top of topic alias yields no further wire-level improvement in this scenario, because in the steady-state workload no messages expire and the publisher count matches the consumer count. (Message expiry would dominate after a prolonged outage, where the publisher's local queue could grow large and stale; shared subscriptions affect the consumer side, which is out of scope for this paper.) Per-message energy falls in step with goodput, from 32,170 μJ (MQTT 5, no extras) to 32,044 μJ (MQTT 5 + alias) per delivered message—a 0.4% reduction.

Summary of Results. The numerical results support three claims that are robust to seed variation:

1. **Steady-state QUIC advantage is negligible on clean links and material only under loss.** At p95 in stable WiFi and 5G conditions, TCP+TLS and QUIC are indistinguishable in the model. At p99 under

3% loss, QUIC saves about 35% (203 vs 311 ms), driven entirely by TCP's 200 ms RTO floor versus QUIC's RTT-proportional PTO.

2. **Mobility-event QUIC advantage depends on the post-handover RTT.** On a 5G→WiFi handover the gain over TCP+TLS 1.3 is small (about 5–10 ms, not statistically significant) because the L2 outage dominates and the post-RTT is only 5 ms; on a 4G→5G handover the gain grows to about 32 ms; on NAT rebinding (no L2 outage) QUIC reduces re-establishment to a single path-validation RTT (about 100 ms saved over TCP+TLS 1.3). Testbed validation suggests these model-based savings are conservative.
3. **Topic aliases dominate the value of MQTT 5** for constrained-payload sensor fleets, lifting goodput efficiency from 0.40 to 0.44 on a 64-byte payload.

Discussion. When QUIC helps. Our model predicts QUIC pays off most for:

- Mobile IoT devices (vehicles, wearables, drones) that change RAT or NAT bindings during a session.
- Lossy radio links (rural cellular, congested ISM-band WiFi) where TCP RTO causes long tail-latency spikes.
- High-QoS telemetry (QoS=2) where the cost of loss is magnified by retry handshakes.
- Frequent reconnects: 0-RTT resumption recovers an idle session without any handshake delay.

When QUIC hurts. QUIC is not free. The model identifies three regimes where staying on TCP+TLS is reasonable:

- Low concurrency, stable network, large payloads. With 4 KB payloads on stable WiFi the goodput-efficiency gap falls to under 2%; the 15% broker-CPU surcharge for QUIC eats the gain.

- QoS=0 at-most-once telemetry. Loss-recovery efficiency is irrelevant because there is no recovery; the only QUIC advantage is reduced handshake cost, which is amortised over a long-lived session.
- NIC offload constraints. Some embedded TCP/IP stacks offload TLS to dedicated silicon; QUIC's user-space crypto is not always supported on resource-constrained MCUs. Our model does not capture this; it is a deployment consideration beyond the scope of this study.

MQTT 5 feature recommendations.

- Always enable topic aliases on small-payload sensor fleets. The 9% goodput improvement and the corresponding energy reduction are essentially free.
- Enable message expiry on devices that buffer locally through outages and reconnect with a queue.
- Use shared subscriptions to scale consumer fleets without a separate queue product.

Limitations. This is a parametric analytical model, not an empirical measurement on a deployed system. The methodology trades fidelity for reproducibility and mechanism isolation; six caveats apply.

The model is a formula calculator, not a packet-level simulator. Each message's latency is computed from a closed-form expression (Eq. 1) with two random draws (Gaussian jitter, Bernoulli loss). The model does not maintain packet queues, flow control state, or congestion windows. The tight confidence intervals in the tables (e.g. ± 0.01 ms on WiFi) reflect the model's low internal randomness, not high measurement precision; a real system would exhibit substantially wider variability from OS scheduling, GC pauses, and congestion dynamics.

QUIC's steady-state advantage comes from one mechanism. In the model, the only mechanism that differentiates QUIC from TCP+TLS in steady-state per-message latency is the retransmission timeout: TCP pays $max(RTT, 200\,ms)$ on the first retransmission whereas QUIC pays RTT. Other potential advantages of QUIC (stream multiplexing, head-of-line-blocking avoidance, more accurate loss detection via monotonic packet numbers) are not modelled. The reported 35% p99 improvement on lossy 4G is therefore a lower bound if these additional mechanisms help, or an upper bound if real-world QUIC implementations introduce offsetting costs (e.g. user-space crypto overhead affecting latency).

Congestion control is not modelled. The model does not model TCP's congestion-window dynamics or QUIC's congestion controllers (NewReno, BBR) explicitly; it models loss recovery as a per-retransmit cost. Our 100-publisher × 256-byte × 10 Hz workload is well below saturation on the modelled uplinks; a heavy-load study would need to extend the model.

NAT-rebinding detection is best-case for TCP. The NAT-rebind result models the reconnection cost as the time to re-establish the transport plus an MQTT CONNECT, assuming immediate detection of the broken tuple. In practice a TCP-based MQTT client typically discovers a broken NAT entry only when the MQTT keepalive PINGREQ goes unanswered—a delay on the order of the keepalive period (commonly 60 s). Our reported figure of about 201 ms for TCP+TLS 1.3 is therefore the lower bound; the real-world QUIC advantage on this scenario is likely much larger.

Broker and energy estimates are constant-factor models. The broker CPU and memory figures are derived from a linear model anchored to EMQX 5.x benchmarks (15% CPU multiplier, 10% memory multiplier for

QUIC), not from live profiling. The per-connection memory model is load-independent (Fig. 5, right panel). The energy model multiplies wire bytes by a per-byte radio-energy constant from Huang et al. [23, pp. 232–233]; it does not capture DRX state transitions or tail timers. Both should be read as order-of-magnitude guidance, not precise predictions.

QUIC testbed validation is pending. We validated the model's TCP+TLS predictions against a Docker-based EMQX testbed (Tables 2–3, the Testbed Validation section). The QUIC predictions could not be validated because no prebuilt NanoMQ Docker image includes msquic-based QUIC transport at the time of writing. Validating the TCP+TLS predictions revealed that the model underestimates p99 on lossy links by about 20% and underestimates handover re-establishment by up to 1.7×. The QUIC absolute numbers should be interpreted with at least the same error margins.

Conclusions and Prospects for Further Research. We have presented, to the best of our knowledge, the first openly published evaluation that jointly considers MQTT 5 features and QUIC connection migration under multi-RAT handover and NAT-rebinding scenarios. Using a parametric analytical model validated against a Docker-based EMQX testbed for TCP+TLS, across 1830 trials spanning six network conditions, three transport stacks, three QoS levels, four payload sizes, three concurrency levels, and four MQTT profiles, the data support three conclusions: (1) QUIC's steady-state advantage over TCP+TLS is negligible on low-loss links and grows to roughly 35% in p99 under 3% loss, driven by TCP's 200 ms RTO floor versus QUIC's RTT-proportional PTO; (2) QUIC's mobility-event advantage tracks the post-handover RTT—small on a 5G→WiFi handover, about 32 ms on a 4G→5G handover, and about 100 ms

on NAT rebinding (where QUIC requires only a single path-validation RTT); testbed validation suggests the model's TCP baseline is optimistic, so the real-world QUIC advantage may be larger; and (3) MQTT 5 topic aliases dominate the protocol-level value of MQTT 5 for small-payload telemetry, improving goodput efficiency from 0.40 to 0.44 on 64-byte payloads.

The model's limitations—no congestion control, no stream multiplexing, unvalidated QUIC predictions, and per-message closed-form computation rather than packet-level simulation—mean that the absolute numbers should be treated as approximate and the relative comparisons interpreted within the error margins established by the testbed validation (up to 1.7× on medians, ~20% on tails).

Future work includes (a) extending the model to a real EMQX + NanoMQ Containernet testbed for a full empirical replication including realistic TCP-failure detection delays driven by MQTT keepalive, (b) characterising 5G-Advanced and 6G handover patterns, (c) studying the interaction of MQTT 5 message expiry with QUIC's stream-level flow control during prolonged outages, and (d) measuring whether QoS=0 telemetry benefits from being carried over QUIC's unreliable datagram extension (RFC 9221 [24]) instead of the default reliable-stream binding.

References:

1. OASIS (2019) *MQTT Version 5.0 OASIS Standard*. Available at: https://docs.oasis-open.org/mqtt/mqtt/v5.0/os/mqtt-v5.0-os.html

2. OASIS (2014) *MQTT Version 3.1.1 OASIS Standard*. Available at: https://docs.oasis-open.org/mqtt/mqtt/v3.1.1/os/mqtt-v3.1.1-os.html

3. Iyengar, J. and Thomson, M. (eds.) (2021) 'QUIC: A UDP-Based Multiplexed and Secure Transport', *RFC 9000*, IETF. Available at: https://www.rfc-editor.org/rfc/rfc9000

4. Thomson, M. and Turner, S. (eds.) (2021) ‘Using TLS to Secure QUIC’, *RFC 9001*, IETF. Available at: https://www.rfc-editor.org/rfc/rfc9001

5. Iyengar, J. and Swett, I. (eds.) (2021) ‘QUIC Loss Detection and Congestion Control’, *RFC 9002*, IETF. Available at: https://www.rfc-editor.org/rfc/rfc9002

6. Bishop, M. (ed.) (2022) ‘HTTP/3’, *RFC 9114*, IETF. Available at: https://www.rfc-editor.org/rfc/rfc9114

7. Duke, M. (2023) ‘QUIC Version 2’, *RFC 9369*, IETF. Available at: https://www.rfc-editor.org/rfc/rfc9369

8. Eddy, W. (ed.) (2022) ‘Transmission Control Protocol (TCP)’, *RFC 9293*, IETF. Available at: https://www.rfc-editor.org/rfc/rfc9293

9. Langley, A., Riddoch, A., Wilk, A., Vicente, A., Krasic, C., Zhang, D., Yang, F., Kouranov, F., Swett, I., Iyengar, J., Bailey, J., Dorfman, J., Roskind, J., Kulik, J., Westin, P., Tenneti, R., Shade, R., Hamilton, R., Vasiliev, V., Chang, W.-T. and Shi, Z. (2017) ‘The QUIC Transport Protocol: Design and Internet-Scale Deployment’, in *Proc. ACM SIGCOMM*, pp. 183–196. Available at: https://doi.org/10.1145/3098822.3098842

10. Hassan, A., Narayanan, A., Zhang, A., Ye, W., Zhu, R., Jin, S., Carpenter, J., Mao, Z.M., Zhang, Z.-L. and Qian, F. (2022) ‘Vivisecting Mobility Management in 5G Cellular Networks’, in *Proc. ACM SIGCOMM*. Available at: https://doi.org/10.1145/3544216.3544217

11. Audet, F. and Jennings, C. (2007) ‘Network Address Translation (NAT) Behavioral Requirements for Unicast UDP’, *RFC 4787*, IETF. Available at: https://www.rfc-editor.org/rfc/rfc4787

12. Kumar, P. and Dezfouli, B. (2019) ‘Implementation and analysis of QUIC for MQTT’, *Computer Networks*, 150, pp. 28–45. Available at: https://doi.org/10.1016/j.comnet.2018.12.012

13. Eggert, L. (2020) ‘Towards Securing the Internet of Things with QUIC’, in *Proc. Workshop on Decentralized IoT Systems and Security (DISS),*

NDSS Symposium. Available at: https://www.ndss-symposium.org/ndss2020/diss-2020/

14. Fernández, F., Zverev, M., Garrido, P., Juárez, J.R., Bilbao, J. and Agüero, R. (2020) 'And QUIC Meets IoT: Performance Assessment of MQTT over QUIC', in *Proc. 16th IEEE Int. Conf. Wireless and Mobile Computing, Networking and Communications (WiMob)*. Available at: https://doi.org/10.1109/WiMob50308.2020.9253384

15. Fernández, F., Zverev, M., Garrido, P., Juárez, J.R., Bilbao, J. and Agüero, R. (2021) 'Even Lower Latency in IIoT: Evaluation of QUIC in Industrial IoT Scenarios', *Sensors*, 21(17), art. 5737. Available at: https://doi.org/10.3390/s21175737

16. Jeddou, S., Fernández, F., Diez, L., Baina, A., Abdallah, N. and Agüero, R. (2022) 'Delay and Energy Consumption of MQTT over QUIC: An Empirical Characterization Using Commercial-Off-The-Shelf Devices', *Sensors*, 22(10), art. 3694. Available at: https://doi.org/10.3390/s22103694

17. EMQ Technologies (2024) *MQTT over QUIC*, EMQX 5.x documentation. Available at: https://docs.emqx.com/en/emqx/latest/mqtt-over-quic/introduction.html

18. EMQ Technologies (2024) *NanoMQ documentation*. Available at: https://nanomq.io/docs/en/latest/

19. De Biasio, A., Chiariotti, F., Polese, M., Zanella, A. and Zorzi, M. (2019) 'A QUIC Implementation for ns-3', in *Proc. Workshop on ns-3*. Available at: https://doi.org/10.1145/3337941.3337945

20. Peuster, M., Karl, H. and van Rossem, S. (2016) 'MeDICINE: Rapid Prototyping of Production-Ready Network Services in Multi-PoP Environments', in *Proc. IEEE NFV-SDN*, pp. 148–153. Available at: https://github.com/containernet/containernet

21. Hemminger, S. (2005) ‘Network Emulation with NetEm’, in *Proc. Linux Conf Au*. Available at: https://man7.org/linux/man-pages/man8/tc-netem.8.html

22. 3GPP (2024) *System architecture for the 5G System (5GS); Stage 2*, TS 23.501, Release 18. Available at: https://www.3gpp.org/dynareport/23501.htm

23. Huang, J., Qian, F., Gerber, A., Mao, Z.M., Sen, S. and Spatscheck, O. (2012) ‘A close examination of performance and power characteristics of 4G LTE networks’, in *Proc. ACM MobiSys*, pp. 225–238. Available at: https://doi.org/10.1145/2307636.2307658

24. Pauly, T., Kinnear, E. and Schinazi, D. (2022) ‘An Unreliable Datagram Extension to QUIC’, *RFC 9221*, IETF. Available at: https://www.rfc-editor.org/rfc/rfc9221

Citation: Yevhen Piotrovskyi (2026). PARAMETRIC EVALUATION OF MQTT 5 OVER QUIC UNDER MOBILITY SCENARIOS. New York. TK Meganom LLC. Innovative Solutions in Modern Science. 3(72). doi: 10.26886/2414-634X.3(72)2026.2