Gossip priorities in Prysm on constrained uplinks

Where a node could prioritise votes, blocks and data columns, and one of those options measured: priority classes in gossipsub's per-peer send queue, on a sixteen-node Kurtosis devnet with uplinks capped between 5 and 38 Mbit/s.

18
runs, 20 minutes each

4
send orders compared

19,887
node-slots measured

Priority can act at several layers, from what is sent to how packets leave the node

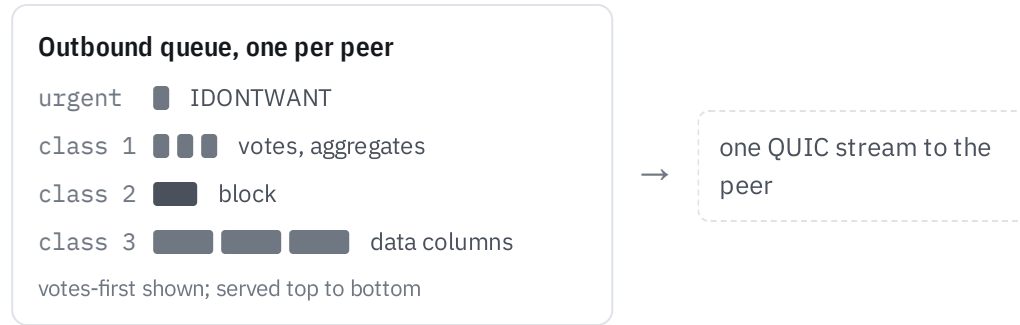
Where	How priority could act	In this study
What is sent gossip protocol	Send fewer redundant bytes, so less has to wait: IDONTWANT, partial messages (cells instead of whole columns), pull instead of push for large topics.	Partial messages compared: off (main runs) and on (slides 11–13); IDONTWANT and pull not varied
Per-peer send queue gossipsub	Order each peer's outbound queue by topic: strict classes, weighted shares, or dropping stale low-priority messages.	**Tested:** strict classes, two orders
Across peers node-wide scheduler	One scheduler above all peer queues, so a higher-priority topic gets the uplink before any peer's lower-priority traffic.	**Tested:** a gate with a bounded hold
Transport QUIC streams	One stream per class, with a priority-aware stream scheduler in the transport.	Not tested: gossipsub uses one stream per peer, and quic-go has no stream priorities
Network and OS qdisc, DSCP	Separate flows per class, prioritised by the kernel's queueing discipline or by DSCP marks along the path.	Not tested: needs one connection per class, since QUIC hides what a packet carries
Timing when to send	Choose when large messages go out: the block before its columns, bursts away from the attestation window.	Partly built in: Prysm already sends the block before its columns

Rows are not exclusive: a node can combine several

Priority classes in gossipsub's per-peer send queue

Picked as a concrete example to measure end to end, not as the best option.

- **Today.** Each peer has one outbound queue of RPCs (Prysm allows 600), sent in arrival order; only IDONTWANT jumps ahead. When the queue is full, new RPCs are dropped.
- **With classes.** Each RPC gets a class from its topic. The queue sends strictly by class, first come first served within a class. When it is full, the newest RPC of the lowest class makes room.
- **Two orders.** `blocks-first`: block › columns › aggregates › votes. `votes-first`: votes and aggregates › block › columns.
- **Variant: across peers.** With `--gossip-uplink-priority`, a peer's writer also holds a lower-class RPC while a higher class is queued for any other peer, for at most 100 ms.



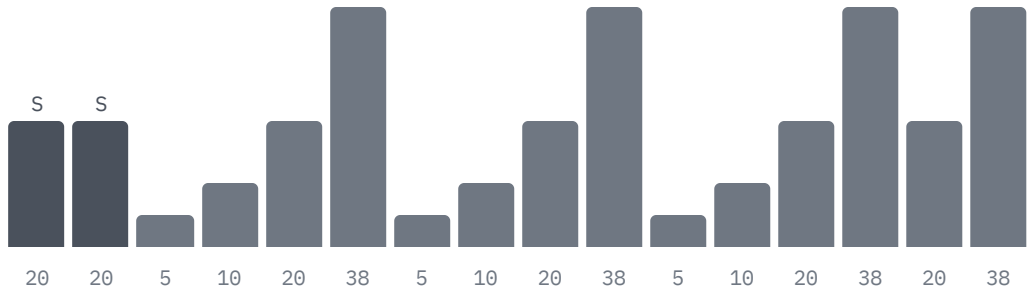
Configurations compared

- Baseline `no priority flag · 3 runs`
- Votes first `--gossip-topic-priority-order=votes-first · 3 runs`
- Votes first, uplink-wide `--gossip-topic-priority-order=votes-first --gossip-uplink-priority · 3 runs`
- Blocks first `--gossip-topic-priority-order=blocks-first · 3 runs`

Sixteen nodes, uplinks from 5 to 38 Mbit/s, the same caps in every run

- **Network.** 16 × geth + Prysm, Fulu (PeerDAS) from genesis, mainnet preset, 12 s slots, 1024 validators. Nodes were asked to keep 8 peers and ended with 6–15, so gossip crosses several hops. The two nodes marked S are supernodes, which store all columns.
- **Load.** spamoor sends blob transactions, so blocks carry data columns, about 90% of all gossip bytes. Main runs with partial messages off (columns travel whole); a second campaign with them on.
- **Links.** Each node's p2p egress (QUIC, UDP 13000) goes through `netem`: 50 ms one way, a per-node rate cap, a 1000-packet queue. The mean cap is about 20 Mbit/s.
- **Runs.** 20 minutes each, the first 30 slots discarded, then about 70 slots per node.

Uplink cap of each node, Mbit/s



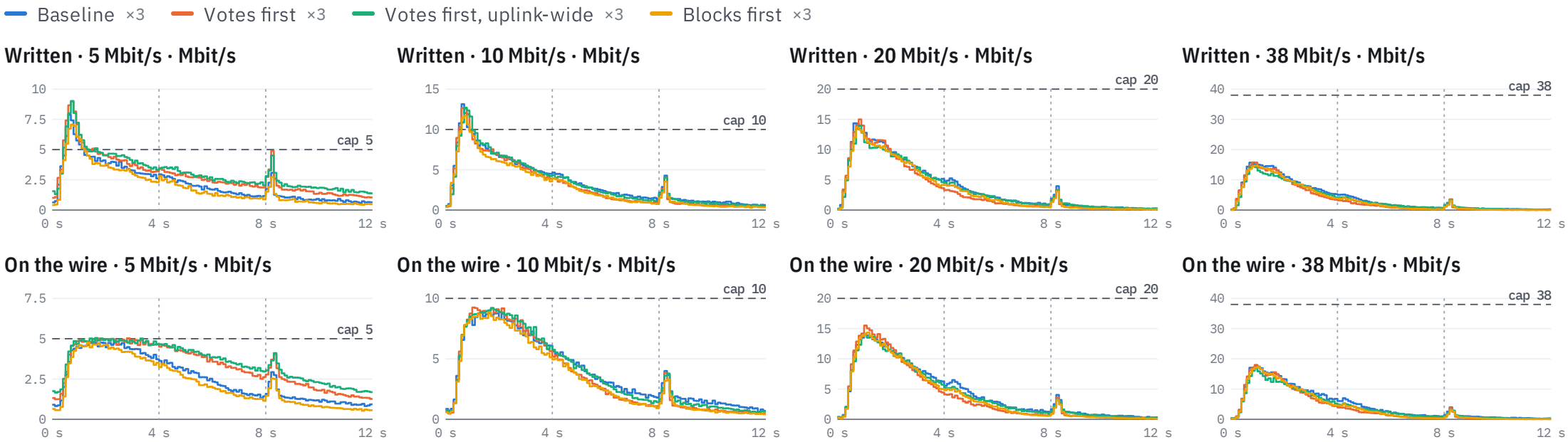
Measured per node and slot

First arrival of each block, column and vote, and block import time.

Per RPC: when it was queued and when its write started and finished, by topic.

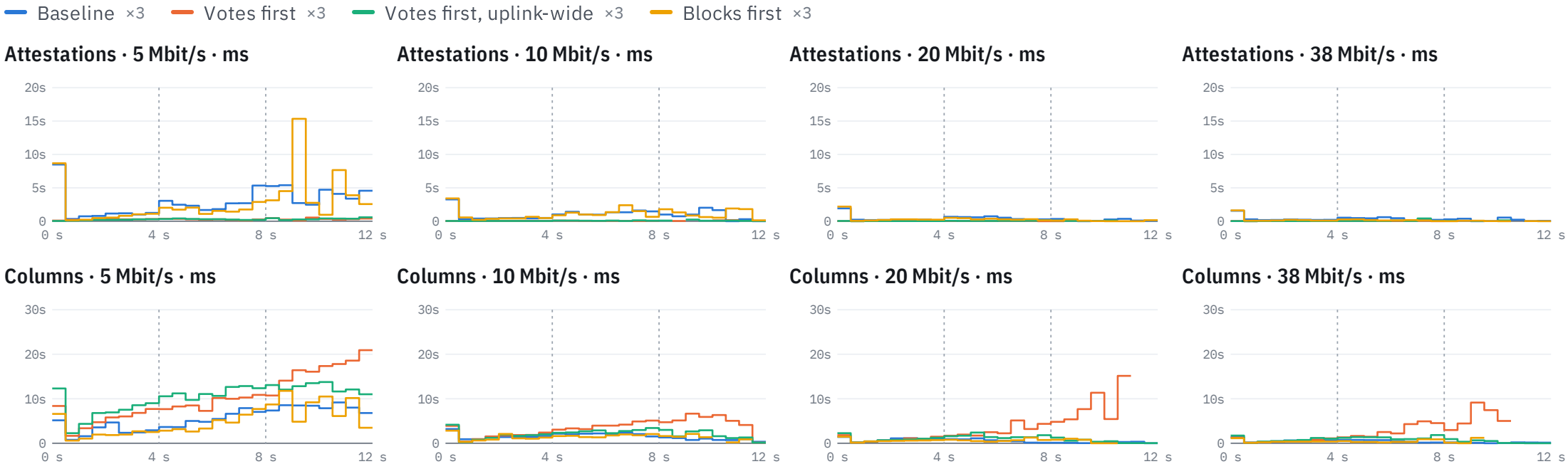
Below the app: the uplink queue's backlog and bytes on the wire, every 100 ms.

Slow uplinks stay saturated for seconds after every block



Per node, all gossip, by time after slot start (100 ms bins, mean over measured slots and the tier's nodes). Top: written into QUIC by gossipsub, which can run ahead of the cap because QUIC buffers. Bottom: bytes actually leaving the node's shaped uplink. Dashed: the cap; dotted verticals: 4 s and 8 s.

Without priority, votes queue behind the column burst

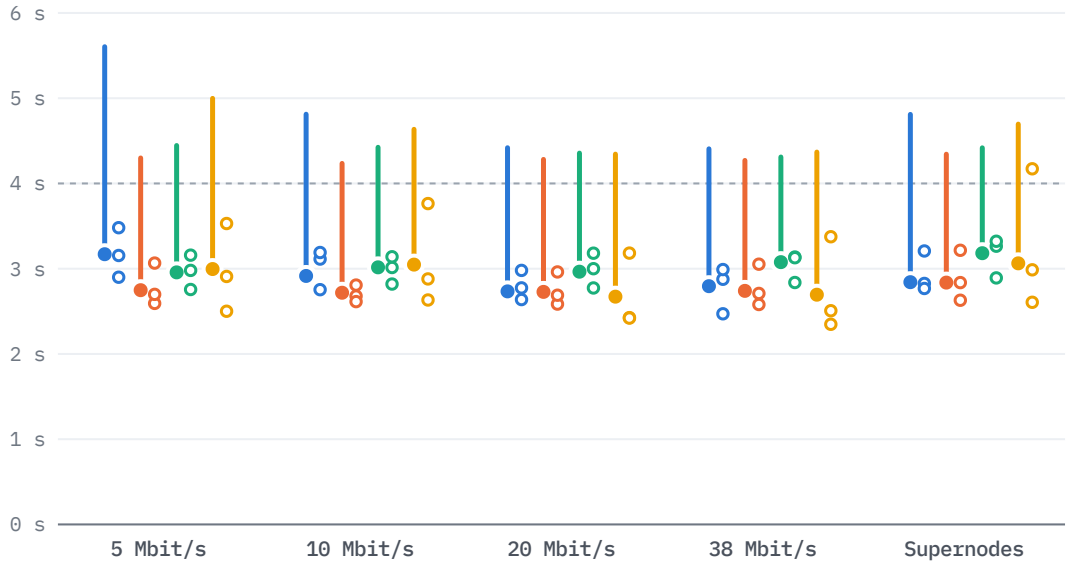


Mean wait in the peer queue of RPCs queued at each moment of the slot. Top row attestations, bottom row data columns, one panel per uplink tier.

Votes first trims the slow-node tail in every run; the median follows block import

Attestation arrival

Dashed: the 4 s attestation deadline.



- Baseline ×3
- Votes first ×3
- Votes first, uplink-wide ×3
- Blocks first ×3

p90 attestation arrival, seconds

Baseline → each other arm, in legend order.

- **5 Mbit/s** 5.60 → 4.30 · 4.45 · 5.00
- **10 Mbit/s** 4.81 → 4.24 · 4.42 · 4.63
- **20 Mbit/s** 4.42 → 4.28 · 4.36 · 4.34
- **38 Mbit/s** 4.41 → 4.27 · 4.31 · 4.37
- **Supernodes** 4.81 → 4.34 · 4.42 · 4.70

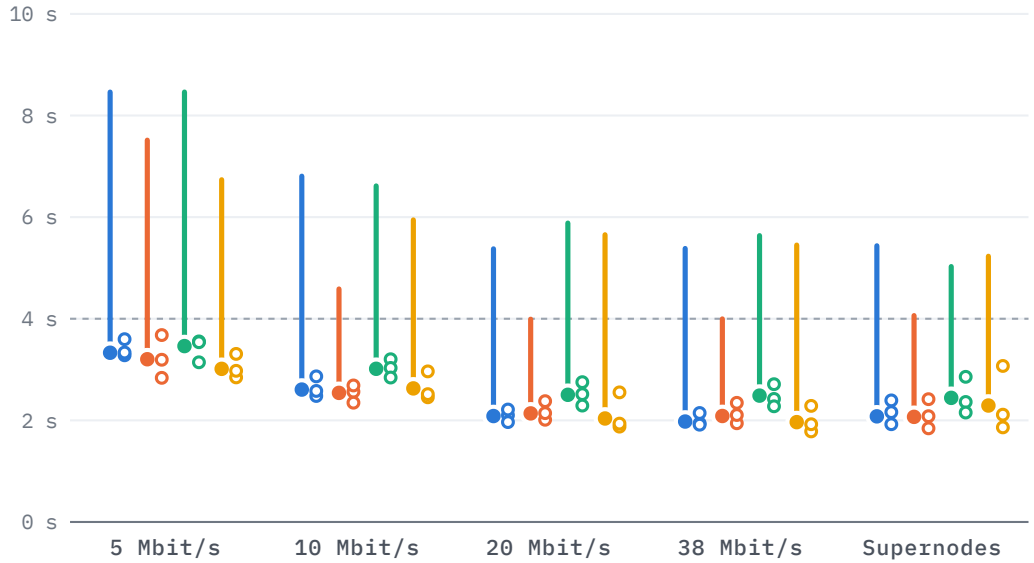
Validators attest once the block is imported, so a vote's arrival is **other nodes' import time plus vote propagation**. Votes first shortens the propagation part (votes wait 82–94 ms in the queue instead of 580–940 ms, per run) so the slow-node tail moves most; the median is set mostly by when blocks are imported, which no order changes reliably.

Median arrival of the slot's attestations at a node; dot p50, bar to p90, pooled over node-slots; small rings: each run's p50

No order reliably speeds up blocks; the uplink gate delays import by about 0.35 s

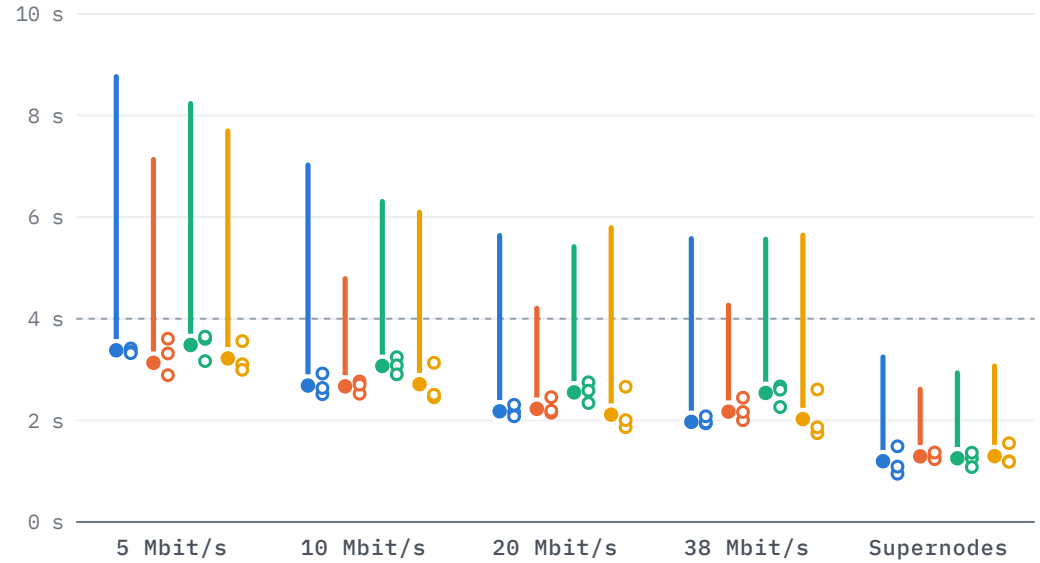
Last data column

Dashed: 4 s after slot start.



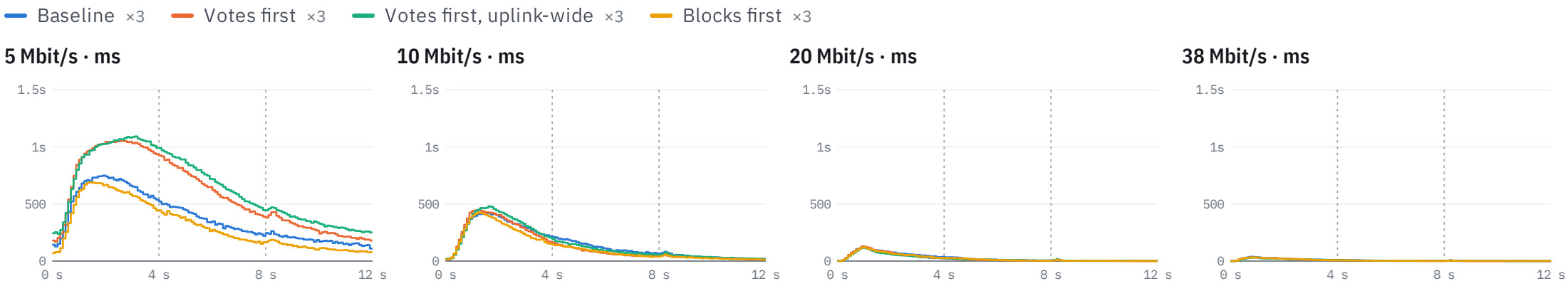
Block import

Dashed: 4 s after slot start.



Proposer's own slots excluded · dot p50, bar to p90, pooled; small rings: each run's p50

After the write, bytes still wait in the uplink queue



uplink	Baseline peak drain · drops per node-slot	Votes first peak drain · drops per node-slot	Votes first, uplink-wide peak drain · drops per node-slot	Blocks first peak drain · drops per node-slot
5 Mbit/s	749 ms · 6.3 741 ms–758 ms · 3.3–9.3	1.05 s · 32.73 1.03 s–1.12 s · 26.3–36.3	1.09 s · 32.86 1.07 s–1.13 s · 21.5–43.1	690 ms · 4.16 621 ms–767 ms · 0.7–9.2
10 Mbit/s	422 ms · 6.07 403 ms–447 ms · 5.6–6.5	448 ms · 4.15 434 ms–507 ms · 3.7–4.7	480 ms · 3.18 438 ms–509 ms · 0.6–6.6	425 ms · 3.58 336 ms–493 ms · 1.6–5.1
20 Mbit/s	116 ms · 0.51 110 ms–126 ms · 0.4–0.6	128 ms · 0.81 106 ms–150 ms · 0.1–2.1	117 ms · 0.34 86 ms–157 ms · 0.0–0.5	120 ms · 0.76 92 ms–152 ms · 0.7–0.8
38 Mbit/s	37 ms · 0.35 38 ms–39 ms · 0.0–0.7	32 ms · 0.28 27 ms–38 ms · 0.0–0.8	32 ms · 0 24 ms–38 ms · 0.0–0.0	33 ms · 0.16 32 ms–39 ms · 0.0–0.3

Shaping-queue backlog ÷ uplink cap, every 100 ms, including the 50 ms delay line; all message types together, since QUIC is encrypted. Table: peak drain · drops per node-slot, pooled; small print: lowest–highest over runs (the first baseline run had no sampler).

Per uplink tier and arm

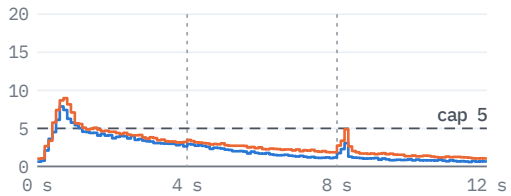
	Baseline		Votes first		Votes first, uplink-wide		Blocks first	
uplink	Attestation	Import	Attestation	Import	Attestation	Import	Attestation	Import
All nodes	2869 / 4583 4478–4733	2199 / 6081 5294–7205	**2739 / 4292** 4176–4398	**2229 / 4791** 4148–6029	3031 / 4389 4273–4486	2594 / 5983 4862–7019	2869 / 4512 4321–4714	2152 / 5870 4921–6617
5 Mbit/s	3170 / 5604 5405–5712	3379 / 8765 7460–9712	**2748 / 4299** 4207–4482	**3133 / 7134** 5882–9295	2957 / 4445 4286–4538	3483 / 8235 6469–9709	2996 / 4998 4545–5942	3219 / 7695 6090–10222
10 Mbit/s	2915 / 4812 4460–5205	2685 / 7026 5123–7710	**2718 / 4235** 3874–4389	**2669 / 4790** 4082–5774	3016 / 4424 4311–4492	3071 / 6308 5028–8155	3048 / 4635 4477–4730	2712 / 6100 5200–7149
20 Mbit/s	2734 / 4419 4392–4528	2178 / 5638 4940–6320	**2728 / 4282** 4157–4421	**2227 / 4207** 3555–5007	2965 / 4356 4218–4470	2551 / 5418 4404–6765	2672 / 4344 4251–4463	2112 / 5790 3877–6639
38 Mbit/s	2795 / 4406 4344–4530	1970 / 5577 4993–6641	**2740 / 4270** 4176–4344	**2171 / 4267** 3670–5450	3077 / 4310 4247–4360	2540 / 5566 4622–7019	2695 / 4368 4220–4498	2025 / 5647 3922–6115
Supernodes	2842 / 4810 4500–5768	1196 / 3247 2751–3947	**2837 / 4343** 4261–4444	**1292 / 2612** 2270–3047	3182 / 4418 4282–4589	1253 / 2935 2331–3311	3063 / 4696 4472–4974	1296 / 3070 2516–3241

Attestation arrival and block import, p50 / p90 in milliseconds after slot start, node-slots pooled over the three runs of each arm; the lowest p90 per row in bold. Small print: the lowest and highest p90 of the single runs. Where two arms' small-print ranges overlap, the difference is within run-to-run spread.

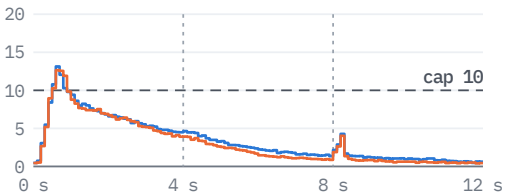
Partial columns barely reduce what nodes send

Baseline `full ×3 · partial ×3` Votes first `full ×3 · partial ×3`

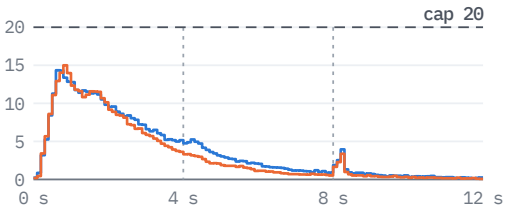
Full · 5 Mbit/s · Mbit/s



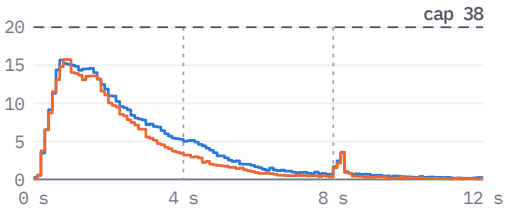
Full · 10 Mbit/s · Mbit/s



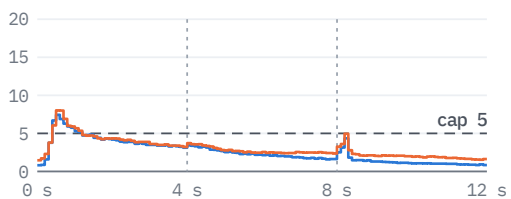
Full · 20 Mbit/s · Mbit/s



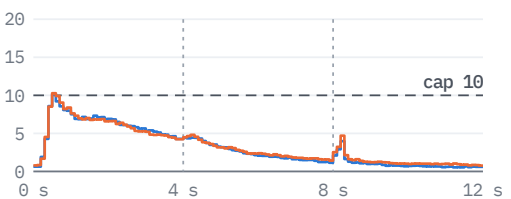
Full · 38 Mbit/s · Mbit/s



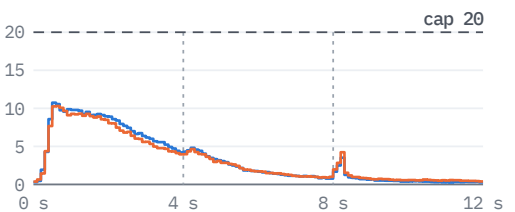
Partial · 5 Mbit/s · Mbit/s



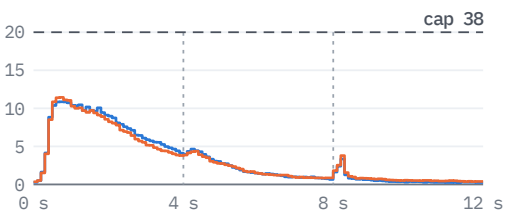
Partial · 10 Mbit/s · Mbit/s



Partial · 20 Mbit/s · Mbit/s



Partial · 38 Mbit/s · Mbit/s

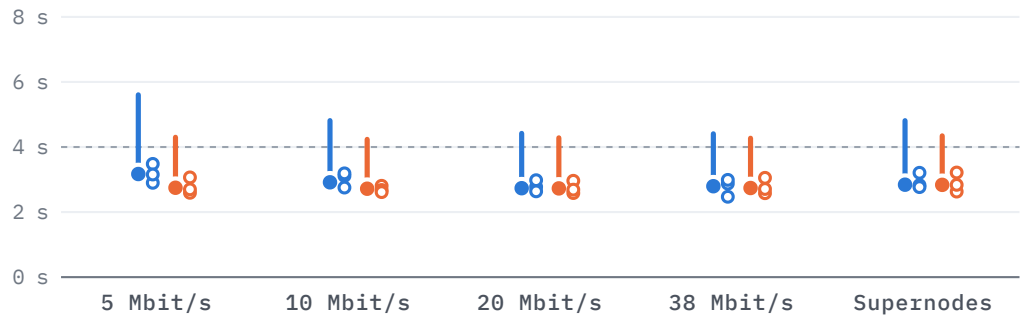


Written into QUIC by gossipsub per node, all topics, mean over node-slots (100 ms bins); dashed: the cap. Column data sent per node per slot: baseline 4.73 → 4.09 MB, votes first 4.22 → 4.21 MB. With partial columns a node completes 98% of its columns from cells sent by peers, 2% from cells it built itself (blobs from its own execution client), 0% as whole columns. Host load was higher during the partial runs (median 1-minute load 113–158 against 51–76 with full columns; the box is shared), so part of any slowdown may be CPU.

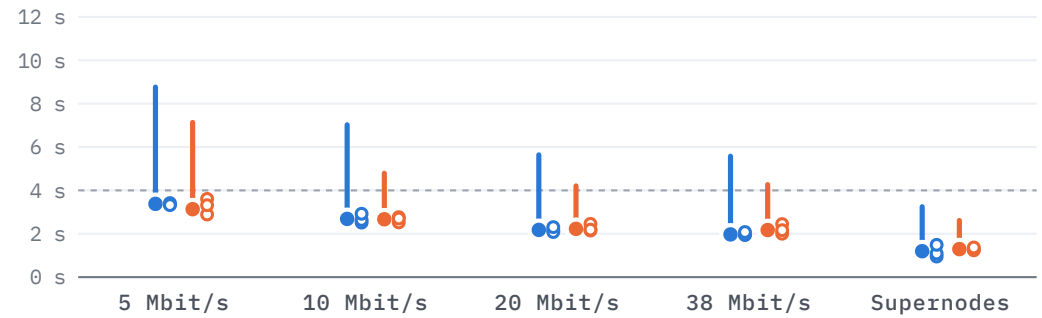
Votes first still trims the slow-node vote tail; with partial columns, blocks import later

Baseline full ×3 · partial ×3 — Votes first full ×3 · partial ×3

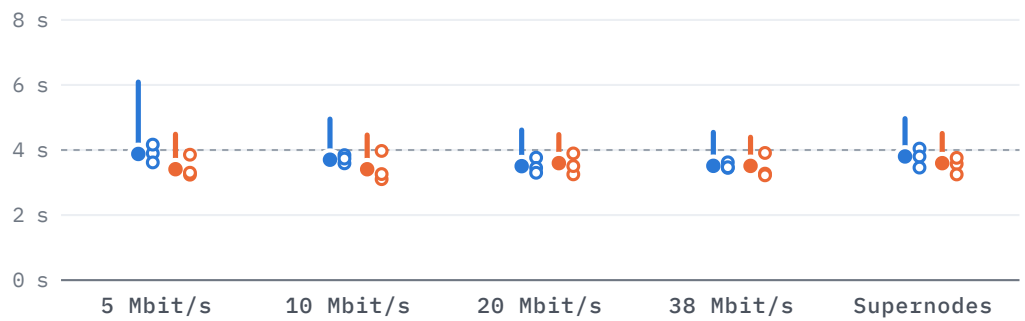
Attestation arrival · full columns



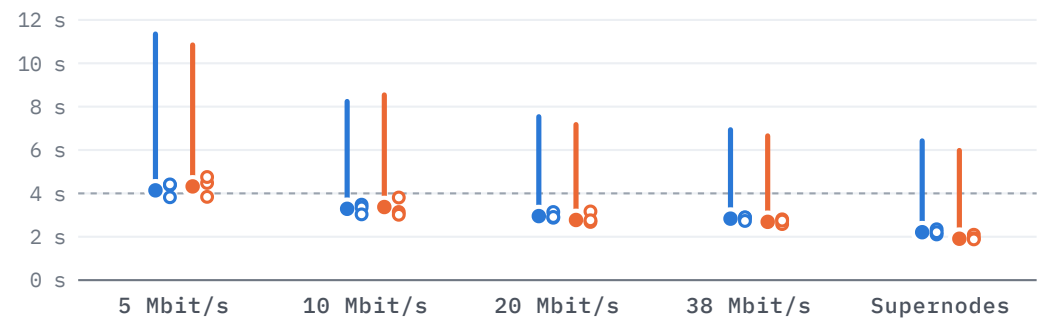
Block import · full columns



Attestation arrival · partial columns



Block import · partial columns



Dot p50, bar to p90, pooled over node-slots; small rings: each run's p50; top row full columns, bottom row partial, same scale

Per uplink tier: full against partial columns

	Baseline full		Votes first full		Baseline partial		Votes first partial	
uplink	Attestation	Import	Attestation	Import	Attestation	Import	Attestation	Import
All nodes	2869 / 4583 4478–4733	2199 / 6081 5294–7205	2739 / 4292 4176–4398	2229 / 4791 4148–6029	3646 / 4853 4711–4938	3061 / 8343 6902–9304	3504 / 4462 4400–4522	2925 / 8493 6894–9473
5 Mbit/s	3170 / 5604 5405–5712	3379 / 8765 7460–9712	2748 / 4299 4207–4482	3133 / 7134 5882–9295	3879 / 6088 5684–6452	4140 / 11349 9960–13708	3410 / 4484 4424–4562	4322 / 10844 9132–12655
10 Mbit/s	2915 / 4812 4460–5205	2685 / 7026 5123–7710	2718 / 4235 3874–4389	2669 / 4790 4082–5774	3702 / 4950 4661–5201	3289 / 8239 6809–9225	3409 / 4460 4400–4501	3371 / 8540 6122–9930
20 Mbit/s	2734 / 4419 4392–4528	2178 / 5638 4940–6320	2728 / 4282 4157–4421	2227 / 4207 3555–5007	3504 / 4614 4481–4691	2953 / 7536 5599–9546	3597 / 4474 4403–4522	2774 / 7172 5531–8753
38 Mbit/s	2795 / 4406 4344–4530	1970 / 5577 4993–6641	2740 / 4270 4176–4344	2171 / 4267 3670–5450	3514 / 4542 4454–4574	2835 / 6935 5873–8797	3512 / 4392 4334–4467	2688 / 6652 5362–7732
Supernodes	2842 / 4810 4500–5768	1196 / 3247 2751–3947	2837 / 4343 4261–4444	1292 / 2612 2270–3047	3802 / 4964 4583–5095	2210 / 6421 5752–8203	3596 / 4508 4477–4534	1909 / 5980 5699–7181
Vote queue wait	772 ms		88 ms		631 ms		65 ms	

Attestation arrival and block import, p50 / p90 in milliseconds after slot start, pooled over node-slots; small print: lowest–highest p90 of the single runs. Last row: mean wait of attestations in the per-peer send queue, all nodes.

Votes first reliably fixes the slow-node vote tail; the uplink gate costs import time; blocks first changes nothing measurable

- **Votes first trims the slow-node vote tail, in every run.** On 5 Mbit/s uplinks, attestation p90 is 4.2–4.5 s in all six runs with votes first (with or without the gate) against 5.4–5.7 s in the three baselines; the ranges do not overlap. Votes stop waiting behind columns: queue wait 580–940 ms → 82–94 ms.
- **Votes first costs blocks nothing measurable.** Import p90 over all nodes is 4.1–6.0 s per run against 5.3–7.2 s for the baseline; if anything lower, but within spread.
- **Blocks first does not change outcomes.** Blocks leave the queue at once (wait 55–197 → 2–3 ms), yet median import is 1.9–2.7 s per run against 2.1–2.3 s: a block is imported when its columns arrive, and they still queue. The earlier median seen in its first run did not repeat.
- **Uplink-wide priority adds nothing for votes and delays import.** Vote tails match votes first alone; median import is 2.4–2.7 s per run against 2.1–2.5 s (pooled +0.35 s, on every non-supernode tier), and votes follow import, so their median is later too. Per-peer priority already gets votes out.
- **0.6–1.1 s of data waits below the app on 5 Mbit/s nodes,** in the uplink queue that QUIC keeps full; gossipsub ordering cannot reach it. Votes first makes it worse: peak drain 0.74–0.76 s → 1.03–1.13 s and drops 3–9 → 21–43 per node per slot, in every run.
- **Partial columns (Prysm's default) do not change the picture.** Votes first still trims the 5 Mbit/s vote tail in every run (4.4–4.6 s against 5.7–6.5 s). Column bytes sent barely fall (4.1–4.2 MB per node-slot against 4.2–4.7 MB with whole columns, of which about 80% are duplicates). Blocks import about 0.8 s later at the median, but the host was more loaded during those runs (median 1-minute load 113–158 against 51–108), so that may be CPU.

Caveats

Caveats

- **Three runs per arm.** Baseline ×3, Votes first ×3, Votes first, uplink-wide ×3, Blocks first ×3. Run-to-run spread is about 0.4 s on median import and up to 2 s on import p90; only differences whose per-run ranges do not overlap are reported as findings.
- **Shared host.** One 40-thread box (2.2 GHz Xeon) shared with other sessions' jobs; median CPU busy during runs up to 77%. Processing delay is part of every latency.
- **Small, random topology.** 6–15 peers per node at the end of runs; who peers with whom changes per run, and each tier has 3–4 nodes.
- **Synthetic load.** spamoor blob traffic and 1024 validators; mainnet has far more attesters per subnet and fewer blobs per node-uplink.
- **Partial columns.** The main runs used `--disable-partial-data-columns`; the partial-columns campaign ran later on a more loaded host (median 1-minute load 113–158 against 51–108), so full against partial mixes protocol and CPU. There, a column counts as arrived when partial cells complete it; supernodes rebuild most columns from half of them, which is not logged, so their column times are not comparable.

Next

- **Separate CPU from protocol** for partial columns: full and partial back to back on a quiet host, with per-container CPU.
- **Drop what the peer no longer wants.** Re-check IDONTWANT when an RPC leaves the queue, and drop columns older than their slot: most of the column backlog is duplicates.
- **Strict-priority QUIC streams.** One stream per class plus a priority scheduler in quic-go, against the uplink gate.
- **Per-type wire timing.** A quic-go hook that reports the stream bytes in each sent packet.
- **A dedicated host and more runs,** then confidence intervals per tier; the block-path effects are smaller than the current spread.