UDK 004.75:519.872:004.056.55

A QUEUEING-DRIVEN CONTROLLER FOR MEMPOOL, BLOCK FULLNESS AND TAIL LATENCY

KHOSHABA O, BYSTRYK M (pzmag2022@gmail.com)

Vinnitsia National Technical University (Ukraine)

Annotation. Permissioned blockchain platforms with Byzantine-fault-tolerant consensus increasingly host latency-sensitive workloads. Yet their performance is governed by coupled, burst-prone subsystems: the transaction mempool, batch (block) formation and leader-based finality. We design, model and evaluate a queueing-driven controller that uses latency quantiles and block utilisation as service-level indicators to steer both mempool policy and block-formation cadence. The modelling view abstracts the mempool and block builder as a finite-buffer batch-service queue calibrated from node telemetry. At the same time, the controller manipulates safe levers — time-bounded adjustments of the block period and conservative changes to admission/eviction in the mempool — guarded by error-budget logic and consensus constraints. On a Hyperledger Besu IBFT/QBFT testbed we observe that, under trace-driven bursts, p99 time-to-inclusion halves from 3.8 s to 1.9 s (-50%) and p95 drops from 2.4 s to 1.6 s (-33%); average block utilisation rises from 82% to 92% while its coefficient of variation halves; sustainable throughput at ρ≈0.9 improves from 270 to 305 TPS without breaching a 2.5 s p99 SLO; mempool drain-time after a burst shrinks from 18 s to 7 s; and IBFT/QBFT round-changes fall from 38 to 11 per 10k blocks. These gains are achieved with ≈1–2% CPU overhead and negligible extra RPC traffic. The results align with queueing-theoretic expectations for finite-buffer batch service and demonstrate a practical path to SLO-centred control in permissioned chains.

Keywords: queueing-based control; Hyperledger Besu; IBFT/QBFT; tail latency; service-level objectives; mempool; block utilisation.

Problem and modelling stance. Latency in permissioned blockchains is dominated by two coupled phases: time spent waiting in the mempool and time to the next successful batch commit. Bursty arrivals, leader variability and bounded blocks induce heavy-tailed delays that make p95/p99 targets fragile [2]. We consider a realistic operating envelope in which the gas limit per block and the block period define the instantaneous service capacity; the mempool enforces admission and eviction; and the consensus layer occasionally triggers round-changes, which transiently reduce effective service. To reason about this system without resorting to protocol-specific minutiae, we adopt a finite-buffer batch-service abstraction whose parameters are calibrated from node telemetry: arrival processes from transaction submission traces; batch-size distribution from gas-accounted inclusion; service cadence from block timestamps; and disturbance processes from observed round-change events. This stance follows classical queueing for computer systems [1] while respecting blockchain particularities such as EIP-1559-style fee pressure [4] and IBFT-style leader rotation [3], [5]. The modelling objective is not closed-form analysis but identifiability of control-relevant sensitivities: how tail latencies and block-utilisation variability respond to small, safe changes in block period and mempool policy, and how these responses shift as load approaches capacity.

Controller design and safety guards. The controller treats user-facing SLOs as first-class: for example, we aim for p99 time-to-inclusion ≤2.5 s and limit the fraction of one-minute windows breaching the SLO to an error budget. Telemetry streamed via Prometheus/Grafana and Besu's JSON-RPC (e.g., txpool_besuStatistics) provides sliding-window estimates of p50/p95/p99 inclusion latencies, average and variance of block utilisation, mempool depth, drop/TTL rates and consensus health (round-changes) [3], [6]. Actions are deliberately conservative. First, a cadence governor proposes small step-changes to the block period within policy bounds, smoothing transitions and avoiding oscillations in leader schedules; such changes trade higher batch frequency

for lower queueing delay when bursts threaten the SLO, but remain bounded to preserve safety margins in consensus processing. Second, a pool governor tweaks admission and eviction: temporarily raising the minimum effective fee, tightening TTL or prioritising transactions that improve packing efficiency. Both mechanisms are wrapped in guards: they respect consensus invariants, rate-limit reconfiguration and default to safe settings under missing telemetry. The two levers are synergistic: cadence reduces waiting time; pool policy improves packing and dampens variance; together they cut tails without destabilising throughput.

Experimental setting and methodology. We evaluate on a Besu network with seven validators (IBFT 2.0 / QBFT), a baseline block period of 2s, and a gas limit of ~15 M. Clients submit a stationary background of simple transfers interleaved with trace-driven Pareto-like bursts 3–5× above background for 2–5s. Windows for SLI computation are one minute (SLO control) and fifteen minutes (stability), with additional roll-ups per 10k blocks for consensus metrics. Baseline runs fix the period and use the default mempool policy, while controlled runs enable both governors with step sizes and bounds pre-vetted for deployment. All measurements are gathered from the RPC and metrics endpoints with identical sampling configurations across conditions [3], [6]. We report medians with interquartile ranges for steady-state metrics and emphasise p95/p99 quantiles for tails, as recommended for user-perceived latency [2]. Where appropriate, we corroborate improvements by showing that mempool peaks shrink and drain faster after a burst, and that the frequency of round-changes decreases, indicating gentler load on the consensus path.

Results. Under the bursty profiles, the controller halves tail latency: p99 inclusion falls from 3.8 s to 1.9 s (-50%) and p95 drops from 2.4 s to 1.6 s (-33%), shrinking the share of one-minute windows breaching the 2.5 s p99 target from 18% to 3%. Average block utilisation increases from 82% to 92% while its coefficient of variation halves, indicating steadier packing; the fraction of near-empty blocks (<30% gas) falls from 7% to 1.5%. At a utilisation ratio around 0.9 of the empirical capacity, sustainable throughput rises from 270 to 305 TPS without violating the p99 SLO, and under mild overload (≈1.1 of capacity) the controller holds a 300 TPS target with ≤20% inflation in p99 and no mempool meltdown. Transient resilience improves: the mempool's peak during bursts decreases from ~20k to ~8k transactions, and the drain-time to background shrinks from 18s to 7s; TTL-based drops decline from 4.2% to 1.3%, reflecting healthier flow. Consensus stability benefits as well: round changes reduce from 38 to 11 per 10k blocks, and the dispersion of block utilisation across validators narrows (e.g., a Gini measure dropping from 0.17 to 0.08), suggesting a fairer proposer workload. Control overheads remain negligible at ≈1–2% CPU and <0.5% extra RPC traffic, with no adverse effects on finality or reorganisation rates. An ablation study clarifies mechanism complementarity: enabling only the pool governor improves p99 by ~22%, while enabling only the cadence governor improves it by ~28%. In comparison, both together reach ~50% — a non-additive gain consistent with reduced variability feeding a faster cadence. These effects are reproducible across IBFT and QBFT variants and align with queueing-theoretic expectations for finite-buffer batch service under bursty arrivals [1], [2], [5].

Discussion, threats to validity and implications. The findings are most directly applicable to permissioned networks where block cadence and mempool policy are configurable and where operational SLOs justify modest, well-guarded adaptivity. Extrapolation to public, fee-auctioned chains requires care: EIP-1559-style dynamics interact with admission rules through base-fee adjustment and priority tips, potentially altering the controller's feasible action set [4]. Our burst traces are representative of enterprise workloads but may not capture adversarial patterns; nevertheless, the reduction in round-changes and in near-empty blocks indicates that the controller reduces stress on consensus and improves packing efficiency in ways that should generalise. Importantly, the controller respects protocol safety by bounding changes, rate-limiting reconfiguration and failing safe on telemetry gaps. From an engineering perspective, the approach encourages operators to frame objectives in terms of quantiles and error budgets, moving beyond averages and peak TPS. From a modelling perspective, the results show that a lightweight, telemetry-calibrated queueing abstraction is sufficient to identify effective levers without invasive

protocol modification. Finally, the small resource footprint and compatibility with stock Besu suggest an attractive cost–benefit trade-off for deployments that must sustain tail-latency SLOs.

Conclusions. We presented a queueing-driven controller that couples mempool policy and block cadence to achieve SLO-centred improvements in latency and stability on a Besu IBFT/QBFT network. Without changing consensus rules, the controller halves p99 time-to-inclusion, raises and steadies block utilisation, increases sustainable throughput near capacity and reduces round-changes, all while imposing minimal overhead. The results substantiate the value of queueing-aware control in permissioned chains and open a path to principled, low-risk adaptivity for latency-sensitive applications. Future work includes integrating fee-aware admission under EIP-1559 economics, extending to smart-contract heavy workloads and exploring formal guarantees for stability under wider perturbations.

References:

[1] M. Harchol-Balter, Performance Modeling and Design of Computer Systems: Queueing Theory in Action. Cambridge, UK: Cambridge University Press, 2013.

[2] J. Dean and L. A. Barroso, “The Tail at Scale,” Communications of the ACM, vol. 56, no. 2, pp. 74–80, 2013.

[3] Hyperledger Besu Documentation, “Consensus protocols and network configuration (incl. IBFT/QBFT),” Available: https://docs.hyperledger.org/besu/latest/ Accessed: 09 Oct. 2025.

[4] T. Harding et al., “EIP-1559: Fee Market Change for ETH 1.0 Chain,” Ethereum Improvement Proposal 1559, Available: https://eips.ethereum.org/EIPS/eip-1559 Accessed: 09 Oct. 2025.

[5] M. Castro and B. Liskov, “Practical Byzantine Fault Tolerance,” in Proc. 3rd USENIX OSDI, 1999.

[6] Hyperledger Besu, “JSON-RPC API Methods (e.g., txpool_besuStatistics, metrics endpoints),” Available: https://docs.hyperledger.org/besu/latest/ Accessed: 09 Oct. 2025.

[7] O. Khoshaba, V. Grechaninov, A. Lopushanskyi, and K. Zavertailo, “Modeling the Process of Loading Impact on Web Servers in Computer Systems,” in Proc. 2021 IEEE 3rd Ukraine Conf. on Electrical and Computer Engineering (UKRCON), Lviv, Ukraine, Aug. 26–28, 2021, pp. 519–524. ISBN: 978-1-6654-0094-7