electronics

MDPI

Article

AdaTrust: Dual-Loop AI-Driven Adaptive Consensus via Bayesian Trust and Deep Reinforcement Learning

Song Tang [1,2,3], Zhigang Jin [1,*] and Zhiqiang Wang [2,3]

[1] School of Electrical and Information Engineering, Tianjin University, Tianjin 300072, China; tangsng@live.cn
[2] Institute of Applied Mathematics, Hebei Academy of Sciences, Shijiazhuang 050081, China; zhiqiangwang19@163.com
[3] Information Security Authentication Technology Innovation Center of Hebei Province, Shijiazhuang 050081, China
* Correspondence: zgjin@tju.edu.cn

Abstract

Blockchain consensus protocols are deployed as static designs in which one protocol with fixed parameters must serve dynamic networks with fluctuating workloads and evolving adversaries. This forces a rigid compromise among throughput, latency, and security. Existing remedies tune a single protocol's parameters or select validators by reputation, leaving the strongest adaptation lever, the protocol identity itself, ungoverned and unproved. Here, we present AdaTrust, a dual-loop adaptive consensus framework. The fast loop, called BayesElect, maintains Beta-posterior trust scores from on-chain votes and elects committees whose size adapts to the estimated Byzantine ratio under a provable corruption bound. The slow loop, called AdaptSwitch, trains a proximal policy optimization (PPO) agent that jointly selects the protocol from an all-BFT pool and its configuration, which is gated by hysteresis. BFT safety is preserved during protocol handover under a bounded-overlap invariant, and policy convergence is characterized under bounded adversarial reward corruption. We evaluate AdaTrust on live blockchains consisting of a 16-validator CometBFT network and a 4-validator Besu QBFT network, with the simulator used only as the controller's training environment. A trust-driven $k = 4$ committee matches full 16-validator Tendermint, achieves 7.8× the matched-workload QBFT throughput, and serves 2.9× the throughput of a reactive threshold switcher; reconfiguration completes in under 8 s, while throughput is flat in committee size (1.9% spread). Under a live silence attack, AdaTrust retains 90.9% of throughput versus 75.9% and heals the committee (k: 4→13) in 6/6 runs; when the Tendermint backend is halted outright, the same controller escapes to QBFT and restores service in 6/6 runs. The results show that the diversity of modern consensus designs can be harvested safely when adaptation is governed by formally bounded trust.

Keywords: blockchain consensus; byzantine fault tolerance; deep reinforcement learning; Bayesian trust management; adaptive consensus; consortium blockchain

Check for updates

Academic Editors: Sara Migliorini, Muhammad Bin Saif and Aysajan Abidin

Received: 30 September 2026
Revised: 16 October 2026
Accepted: 19 October 2026
Published: 22 October 2026

Copyright: © 2026 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license.

1. Introduction

1.1. Research Background and Motivation

The value proposition of a blockchain—reaching agreement among parties that do not trust one another—rests entirely on its consensus layer, the mechanism that imposes order and finality on transactions [1]. Beginning with Nakamoto's Proof-of-Work and the original PBFT construction of Castro and Liskov [2], the field has produced a wide menu of designs: leader-driven BFT such as HotStuff [3], stake-based families [4], DAG-structured protocols

like Narwhal/Tusk [5] and Bullshark [6], network-adaptive systems such as Jolteon and Ditto [7], and asynchronous alternatives, exemplified by Dumbo-NG [8]. Beyond consensus itself, the security of blockchain-anchored systems rests on complementary cryptographic services beneath the consensus layer, from efficient group authentication and key agreement with non-linkability for industrial IoT deployments [9] to quantum-resistant certificateless signcryption for blockchain-based smart healthcare [10]. All of these assume a functioning consensus substrate, which is the layer we target in this work.

Despite this breadth, deployed systems share a single telling shortcoming in that the protocol and its parameters are frozen at launch. The configuration is chosen once, then must cope with conditions its designers could only guess at. Three issues make this untenable: first, network state is never stationary, as propagation delay, bandwidth, and membership churn all drift [11]; second, workload is bursty, meaning that the offered load commonly changes by an order of magnitude within hours, so any single block size or timeout setting is wrong for either the peak or the trough; third, adversaries can adapt by studying the protocol's fixed attack surface and accordingly targeting leader denial-of-service, equivocation, or strategic silence [12]. Impossibility and trade-off results confine any individual design point [1]; however, nothing stops a system from moving across that space as conditions change. This possibility has already been exploited by adaptive replication systems in the broader literature on distributed systems [13], and is the point of departure for our work.

Two lines of research have begun to relax the static assumption. One brings deep reinforcement learning (DRL) [14–16] to the task of tuning a single protocol's parameters: DA-PBFT for PBFT in IoT settings [17], DAGWise for GNN-driven prediction of batching and quorum choices [18], DESAC for vehicular consensus [19], R-MADRL for leader election and wait windows [20], and validation-based RL tuning more generally [21]. The other line governs who participates, concentrating consensus among validators with strong reliability records through reputation- or trust-weighted committee election [11,22–29].

1.2. Problem Statement

Each line on its own misses the decisive lever. Parameter tuning never leaves its protocol family: no PBFT configuration can reproduce HotStuff's linear authenticator cost, and no HotStuff setting delivers PBFT's immediate finality—protocol identity is the strongest available degree of freedom, and it is precisely the one left untouched. Trust-based election, conversely, decides membership but not behavior: even a perfectly chosen committee continues to run a fixed protocol that may be poorly matched to the moment. What production deployments actually need is absent from both: (i) a formal guarantee that learned or trust-based selection cannot be steered into an unsafe state, and (ii) a safety argument that holds across the act of adaptation itself—the window in which one protocol yields to another.

We address the following problem: construct a consensus architecture that (a) adapts committee membership and protocol identity (with parameters) to observed network, workload, and adversarial conditions; (b) bounds the probability of an adversary-controlled committee; (c) preserves BFT safety throughout every switch; and (d) converges under partial observability and adaptive adversaries.

1.3. Limitations of Existing Methods

Four specific gaps, elaborated in Section 2, motivate the design.

- **L1—Confinement to one protocol.** DA-PBFT [17], DAGWise [18], DESAC [19], R-MADRL [20], and validation-driven tuning [21] optimize within a single family; the protocol choice is never an action. The few systems that do treat identity as a decision

variable—Abstract [30] (a predefined abort order), ADAPT [31] (supervised selection), BFTBrain [32] (a contextual bandit)—leave membership ungoverned and offer no formal treatment of committee corruption or of the transition itself (L3–L4).

- **L2—Trust without a sizing rule.** rPoA [22] and sharding committee schemes [11] pick members by reputation, but size the committee by convention rather than against a corruption probability; WBR [23] is Bayesian, yet targets transaction classification, not committee formation; and hierarchical BFT variants [26–28] retain heuristic sizing.
- **L3—No analysis of the transition.** Adaptive systems [17–21] report steady-state throughput but say nothing about the reconfiguration window—view changes, in-flight quorums, divergent state—which is exactly where liveness and safety break. Recent mechanized proofs for DAG consensus [33] make the point that transition safety must be demonstrated, not assumed.
- **L4—No convergence characterization.** DRL-based proposals publish empirical learning curves but state neither convergence conditions nor a bound relative to the clairvoyant optimum, a gap that matters most when the reward signal itself is partially under adversarial control [12].

1.4. Contributions

We propose **AdaTrust**, a dual-loop design in which a Bayesian trust layer decides who runs consensus (a fast loop) and a DRL controller decides which protocol and parameters they run (a slow loop), both wrapped in formal guarantees.

1. **BayesElect: Bayesian trust-driven adaptive committee election.** Each validator carries a Beta posterior $T_i = \alpha_i/(\alpha_i + \beta_i)$ maintained from on-chain votes with exponential decay. From this, we derive a closed-form committee size $k^*(n, \hat{f}, \epsilon)$ that keeps the corruption probability below a configurable ϵ (Section 3.2.2). On the live chain, a crashed validator's trust crosses the detection threshold in ≈53 blocks (consistent with the decay theory), the committee self-heals (k 4→13) in 6/6 attack runs, and throughput is flat in k (1.9% spread), so the safety margin of Theorem 1 is obtained at zero throughput cost (Sections 4.2 and 4.3).
2. **AdaptSwitch: DRL-driven joint protocol-and-parameter switching.** Cross-protocol adaptation is cast as a Markov decision process in which the action space jointly selects a protocol from an all-BFT pool and its configuration, gated by hysteresis (Sections 3.2.3 and 3.2.4). On the measured pool (Section 4.1), the deployed policy matches full-set Tendermint at $k = 4$ (653.7 TPS), reaches 7.8× the matched-workload QBFT throughput, and correctly avoids switches: the clairvoyant per-phase-best schedule degenerates to no-switch, while a reactive switcher forfeits 65.7% of throughput. Its QBFT legs are ramp-limited to ≈20 TPS effective, never reaching the 84.3 TPS steady state, and drain-containing switch epochs consume ≈35% of wall time. An offline ablation shows the switching cost term stops the policy from collapsing into a static corner during learning (Section 4.4). When the Tendermint backend itself is halted (6 of 16 validators stopped, below quorum), the same policy reverses course and escapes to QBFT in 6/6 paired runs (95.1 ± 5.6 s), restoring degraded but nonzero service where the static chain serves zero (Section 4.5).
3. **Formal safety and convergence analysis.** We prove (i) a committee-corruption bound under BayesElect; (ii) preservation of BFT safety across protocol handover under a bounded-overlap invariant (a design-time guarantee); and (iii) convergence conditions together with bounds on hysteresis-induced suboptimality and on adversarial reward corruption (Section 3.3).
4. **Real-chain reproducible evaluation.** On a dual-stack testbed (16-container CometBFT v0.38.17 plus a 4-node Besu 24.12.2 QBFT network, with the simulator used only for

training), AdaTrust is compared against static Tendermint, matched-workload QBFT, and a reactive switcher in interleaved paired runs (Sections 3.4 and 4). Under a live silence attack it retains 90.9% of pre-attack throughput against the static chain's 75.9%.

Gap statement. Protocol switching is not new, as Abstract [30], ADAPT [31], and BFTBrain [32] each switch in some form; to the best of our knowledge, however, AdaTrust is the first architecture to unify all three of the following: (i) treating protocol identity as a learnable resource through deep RL over a factored action space spanning identity, committee size, and block parameters rather than a fixed order, supervised selector, or bandit over identities alone; (ii) coupling that controller to a Bayesian trust layer with a provable corruption bound that adapts committee membership and size; and (iii) guaranteeing safety not only in steady state but throughout the adaptation transitions themselves, together with convergence and robustness bounds for the learning process. Prior work attains at most one of these in isolation (Table 1).

Table 1. Comparison with representative works. N/A: not applicable. The row in bold highlights the proposed system (AdaTrust).

Work	Year	Adaptation Target	AI Method	Adversary Model	Formal Guarantee	Switching Safety
PBFT [2]	2002	None (static)	—	Static Byzantine, $f < n/3$	Safety/liveness proofs	N/A
HotStuff [3]	2019	None (static)	—	Static Byzantine	Linear view change	N/A
Jolteon & Ditto [7]	2022	Sync/async fallback	—	Network adversary	Protocol proofs	N/A
Bullshark [6]	2022	None (DAG-BFT)	—	Static Byzantine	DAG-BFT proofs	N/A
Dumbo-NG [8]	2022	None (async)	—	Network adversary	Async broadcast proofs	N/A
SquirRL [12]	2021	(attacker's view)	DRL	Rational miner	—(analysis tool)	N/A
RepChain [24]	2021	Committee (reputation+sharding)	—	Static Byzantine	Incentive compatibility	N/A
rPoA [22]	2024	Committee (reputation)	—	Static Byzantine	Heuristic fork-choice	×
WBR [23]	2024	Trust weighting (classification)	Bayesian	>50		
DA-PBFT [17]	2024	Parameters (PBFT only)	Multi-agent DRL	Incomplete information	Nash/incentive analysis	×
DAGWise [18]	2025	Parameters (DAG-BFT only)	GNN	Static-parameter degradation	Invariant-preserving integration	×
DESAC [19]	2025	Parameters + trust weights	Diffusion SAC	Trust abuse	Empirical	×
Abstract [30]	2015	Protocol identity (abortable composition)	Predefined order (no learning)	Byzantine (via BFT protocols)	×	× (fall-through on abort)
ADAPT [31]	2015	Protocol identity (BFT pool)	Supervised prediction	Byzantine (via BFT protocols)	×	× (abort-and-replace)
BFTBrain [32]	2025	Protocol identity (BFT pool)	Contextual bandit	Byzantine (via BFT protocols)	× (empirical)	× (relies on coordination primitive)
AdaTrust (this work)	—	**Committee + protocol identity + parameters**	**PPO + Bayesian posterior**	**Static and adaptive Byzantine, collusion, switching-aware**	**Committee corruption bound + convergence conditions**	**✓ (bounded-overlap invariant)**

2. Related Work

2.1. Evolution of Consensus Protocols: From PoW to BFT Families

Permissionless consensus begins with Proof-of-Work [34], which spends energy on Sybil resistance in exchange for only probabilistic finality [1]. Proof-of-Stake substitutes stake for energy, securing accountable safety at the price of delayed finality [4]. The partial synchrony framework that underlies modern BFT traces back to Dwork, Lynch, and Stockmeyer [35]. DAG-based designs such as Narwhal/Tusk [5], Bullshark [6], and their analyzed successors [36,37] decouple data dissemination from ordering to lift throughput, while network-adaptive protocols such as Jolteon and Ditto [7] add asynchronous fallback. In the permissioned and consortium settings that we target here, BFT-style voting protocols prevail: PBFT [2] offers immediate finality at $O(n^2)$ messages per view, HotStuff [3] cuts authenticator cost to linear using threshold signatures and pipelining, Tendermint [38] trades gossip-based vote dissemination for $O(n \log n)$ messages and immediate finality, and asynchronous protocols such as Dumbo-NG [8] drop timing assumptions at the cost of latency. Raft [39] covers crash-only faults, sitting at the low-latency end of the spectrum. Sharding splits validators to scale throughput, but pays cross-shard coordination overhead and per-shard security dilution [11].

Open problems. Every protocol above occupies one fixed point in the throughput–latency–security–assumption space. The literature keeps extending the menu, yet a deployment almost always orders a single item and never revisits the choice. How to change menu items safely at runtime remains open.

2.2. Adaptive Consensus Mechanisms

Adaptation has been explored at three granularities. Parameter adaptation tunes block size, timeout, or quorum within one protocol; DAGWise predicts batching delays, quorum thresholds, anchor sets, and commit scores from certified DAG snapshots via GNN heads [18]; R-MADRL adapts leader election and consensus wait windows for end–edge computing [20]; and validation-driven RL adjusts consensus parameters [21]. Committee adaptation changes the participant set: Algorand samples committees by stake with verifiable randomness [40]; rPoA lets nodes self-organize committees from reputation variation [22]; sharding schemes rebalance membership by reputation [11]; and reputation-enhanced election biases proposer choice [29]. Protocol adaptation, in which the protocol itself is switched, is the least explored, with two precursors predating learned control. Abstract [30] treats BFT instances as abortable units; on stalled progress, it aborts and falls through a predefined order that is fixed in advance rather than being learned. ADAPT [31] replaces that order with an evaluation step that matches protocol profiles to operator preferences via supervised prediction, though it still needs a predictor per protocol–condition pair and offers no guarantee on the transition. Elsewhere in distributed systems, AWARE adapts wide-area replication through learned voting and leader tuning [13], but stops short of cross-protocol switching. The closest contemporary work is BFTBrain [32], which learns runtime switching among BFT protocols: it models per-condition performance with a contextual multi-armed bandit, then coordinates the learned output across replicas through a validated broadcast primitive, reporting throughput gains of 18–119% on a SmashBFT-derived testbed. BFTBrain shows the protocol identity lever is pullable in practice; it leaves open the governance and guarantee dimension: full validator participation throughout (no trust-weighted committee, so no corruption bound), safety that rests on the correctness of a coordination primitive rather than a proved handover invariant, and no convergence or suboptimality characterization under adversarial reward influence.

Open problems. Existing mechanisms either stay within one protocol family or, BFTBrain aside, switch identity without membership governance; they lack a formal treatment

of the transition interval in which configuration c_1 is retired and c_2 activated, and none coordinates parameter-, protocol-, and membership-level adaptation together. AdaTrust's dual loop targets all three gaps at once.

2.3. AI/ML and Trust Management in Blockchains

Learning has been applied across the stack [15,16]: consensus optimization (DA-PBFT's multi-agent DRL for PBFT in IoT [17], DESAC's diffusion-enhanced soft actor–critic for platoons [19]), attack discovery, where SquirRL shows that DRL can find profitable attacks on incentive mechanisms [12], and incentive design plus blockchain-secured federated learning [41–44]. PPO [14] is used to anchor our controller due to its stability under joint discrete–continuous actions [18,20]. On the trust side, Bayesian reputation models update Beta posteriors from binary evidence [45,46]; within blockchains, rPoA couples off-chain reputation to on-chain verification [22], WBR applies Bayesian weighting to IoT transaction classification [23], RepChain fuses reputation with sharding [24], hierarchical schemes improve PBFT efficiency [25–28], and trustworthy reputation games formalize endorsement-based selection with Nash equilibrium decoding [47]. Closest to our participation-governing use of trust is Pawar et al. [48], who combined a PBFT mechanism with global reputation reporting. In this approach, the weights of impacted nodes are corrected by high-trust measures until a Bayesian equilibrium is reached in IoT secure storage and trusted data sharing; there, reputation disciplines data sharing, whereas in AdaTrust trust directly governs consensus participation with a derived committee size and a corruption bound.

Open problems. Three shortcomings recur across both threads: (i) learning addresses parameters, never protocol identity; (ii) convergence claims are empirical, with no stated conditions or bounds against a clairvoyant switching oracle; (iii) trust scores are heuristic aggregates lacking a derived committee size tied to a corruption probability, their dynamics (decay, slashing) are seldom modeled for formal manipulation-resistance, and the trust layer is never co-designed with protocol selection. AdaTrust's dual loop closes all three.

2.4. Summary and Positioning

Table 1 positions AdaTrust against representative systems, comparing their adaptation target, learning method, adversary model, formal guarantees, and switching safety. The pattern is uniform, with each prior line securing one layer while leaving the others static. AdaTrust's contribution is the joint adaptation of membership (BayesElect) and protocol identity-plus-parameters (AdaptSwitch), wrapped in proofs that also hold during adaptation, not just after.

3. Materials and Methods

3.1. System Model and Problem Formulation

3.1.1. Network Model

We consider a permissioned blockchain with a validator set $\mathcal{N} = \{1, 2, \ldots, n\}$, where n is known and identities are bound to public keys registered on-chain (no Sybil identities). Validators communicate over authenticated point-to-point channels; messages are signed and non-forgeable. Time is divided into epochs $e = 1, 2, \ldots$, each comprising E consensus rounds (blocks).

We adopt the partial synchrony model [1]: there exists an unknown Global Stabilization Time (GST) and a known bound Δ such that every message sent after GST is delivered within Δ. Before GST and during adversarially induced degradation, the one-way delay d_{ij} between validators i and j follows a heavy-tailed distribution, modeled as log-normal, $d_{ij} \sim \text{LogNormal}(\mu_{ij}, \sigma_{ij}^2)$, the parameters of which drift over time. The available bandwidth $b_{ij}(t)$ and the transaction arrival rate $\lambda(t)$ are time-varying exogenous processes; $\lambda(t)$ is

driven in our experiments by wall clock square-wave workloads (Section 3.4.5). This model's ability to realistically capture events such as load bursts, delay spikes, and regime shifts represents an improvement on static protocol configurations and motivates the adaptation in this paper.

3.1.2. Threat Model

At any epoch e, at most f validators are Byzantine—with $f < n/3$ globally—and the elected consensus committee must contain fewer than one-third Byzantine members for BFT safety. Byzantine validators may deviate arbitrarily. Below, we enumerate the attack classes considered:

- **A1—Equivocation**: Conflicting votes/proposals for the same height or view.
- **A2—Strategic silence (on–off attack)**: A validator behaves honestly to accumulate trust but withholds votes when pivotal, degrading liveness.
- **A3—Collusion**: A coalition of size $c \leq f$ coordinates equivocation/silence to maximize disruption and evade per-node detection.
- **A4—Switching-aware attack**: The adversary observes protocol switch signals and concentrates attacks during handover windows, where transient inconsistency is most likely.

We distinguish adversary schedules into static (corruptions fixed at genesis), adaptive (the adversary re-selects its corruption targets at each epoch based on the observed protocol and committee), and mobile (corruption sets change over time subject to a per-epoch budget). The adversary controls the network subject to the partial synchrony bound after GST, but cannot break standard cryptographic primitives.

3.1.3. Trust Model

Each validator i carries a Beta posterior over its per-action honesty probability θ_i, with hyperparameters (α_i, β_i) initialized to $(1, 1)$ (uniform prior). The trust score is the posterior mean:

$$T_i = \frac{\alpha_i}{\alpha_i + \beta_i}, \qquad \mathrm{Var}[\theta_i] = \frac{\alpha_i \beta_i}{(\alpha_i + \beta_i)^2(\alpha_i + \beta_i + 1)}. \tag{1}$$

Evidence is drawn from objectively verifiable on-chain behavior at block granularity: for each block, a correct timely vote yields $\alpha_i \leftarrow \alpha_i + 1$, an incorrect or missing vote yields $\beta_i \leftarrow \beta_i + 1$, and a cryptographic proof of equivocation yields a slash, $\beta_i \leftarrow \beta_i + \kappa$ with $\kappa \gg 1$. To bound the lifetime of stale evidence and defend against on–off attacks (A2), hyperparameters decay exponentially per block with factor $\lambda_d \in (0, 1)$:

$$\alpha_i \leftarrow 1 + \lambda_d(\alpha_i - 1), \qquad \beta_i \leftarrow 1 + \lambda_d(\beta_i - 1). \tag{2}$$

Thus, trust reflects a recency-weighted history; a validator must continuously behave in order to retain a high score. The control loop of Section 3.2 acts at coarser epoch granularity in which an epoch is a fixed number of blocks, while trust evidence and decay are updated per block; the deployed system of Section 3.4 inherits this per-block update rule, making the measured detection latency of Section 4.3 directly comparable to the decay dynamics of Equation (2). Trust-weighted sampling assigns selection weight $w_i = T_i^{\eta}$ to validator i, where $\eta \geq 1$ sharpens the preference for high-trust validators. All trust evidence is derived from data that are already present on-chain; no additional reporting channel is required [45,46].

3.1.4. Consensus Protocol Abstraction

The controller operates over an all-BFT protocol pool $\mathcal{P} = \{\mathrm{PBFT}, \mathrm{HotStuff}, \mathrm{Tendermint}\}$, which is chosen to span three distinct operating niches relevant to consortium deployments:

PBFT [2] offers the lowest normal-case latency at small-to-moderate committees (three message delays, immediate finality) but quadratic messages and $O(k)$ per-node verification. HotStuff [3] carries one extra pipeline delay (three-chain finality) in exchange for $O(1)$ aggregated verification and linear messages, dominating at large committees. Finally, Tendermint [38] matches PBFT's voting-phase latency and immediate finality with $O(k \log k)$ gossip messages; its block-part gossip overlaps proposal dissemination with voting, raising effective block capacity under high offered load at the price of a store-and-forward dissemination delay that grows with block size. We deliberately exclude crash-fault tolerant (CFT) protocols such as Raft: under the Byzantine threat model of Section 3.1.2, a CFT protocol has no valid operating point; a single equivocating leader can fork a CFT chain irrevocably, so including one would expose the learned controller to a structurally unsafe action. Restricting the pool to protocols that each tolerate $\bar{f}_p = 1/3$ makes every protocol switching action meaningful in terms of safety, while the three niches (latency/scalability/throughput) keep the switching lever materially useful. Each protocol p is characterized by a feature vector (Equation (3)):

$$\boldsymbol{\phi}(p) = \left(c_p^{\mathrm{msg}},\ \delta_p^{\mathrm{fin}},\ \ell_p,\ \bar{f}_p\right) \tag{3}$$

where $c_p^{\mathrm{msg}} \in \{O(k), O(k \log k), O(k^2)\}$ for committee size k, δ_p^{fin} is the number of blocks to finality, $\ell_p \in \{0, 1\}$ indicates leader-based operation, and $\bar{f}_p = 1/3$ is the maximum tolerable Byzantine ratio for every $p \in \mathcal{P}$. Every $p \in \mathcal{P}$ exposes a uniform parameter interface $\pi_p =$ (block size B, timeout τ), and is assumed to be individually safe and live under the partial synchrony assumption of Section 3.1.1.

3.1.5. Problem Formulation: Adaptive Consensus as an MDP

We cast cross-epoch consensus adaptation as a Markov decision process $\mathcal{M} = (\mathcal{S}, \mathcal{A}, P, R, \gamma)$, defined at epoch granularity.

State space $\mathcal{S}$**.** The state at epoch t aggregates observable system descriptors

$$s_t = \left(\mathbf{w}_t,\ \mathbf{u}_t,\ \mathbf{g}_t,\ \mathbf{h}_t,\ \mathrm{enc}(p_{t-1})\right) \in \mathcal{S}, \tag{4}$$

where $\mathbf{w}_t = (\hat{\lambda}_t, \bar{s}_t^{\mathrm{tx}}, \hat{\mu}_t^d, \hat{\sigma}_t^d)$ summarizes workload and network statistics, $\mathbf{u}_t = (\bar{T}_t, \sigma_T^2, q_{0.1}(\mathbf{T}_t), \hat{f}_t)$ summarizes the trust landscape, $\mathbf{g}_t \in [0, 1]^3$ collects attack signals, $\mathbf{h}_t = (\widetilde{\mathrm{TPS}}_{t-w:t}, \hat{L}_{t-w:t})$ contains the moving window performance history, and $\mathrm{enc}(p_{t-1})$ is a one-hot encoding of the currently active protocol.

Action space $\mathcal{A}$**.** Actions are joint protocol and configuration decisions:

$$a_t = \left(p_t, B_t, \tau_t, k_t\right) \in \mathcal{A} = \mathcal{P} \times \mathcal{B} \times \mathcal{T} \times \mathcal{K} \tag{5}$$

with p_t the selected protocol, $B_t \in \mathcal{B}$ the block size (discretized), $\tau_t \in \mathcal{T}$ the timeout, and $k_t \in \mathcal{K}$ the committee size hint passed to BayesElect (subject to the safety floor $k_t \geq k^*$ derived in Section 3.2.2).

Transition kernel P**.** $P(s_{t+1} \mid s_t, a_t)$ is unknown and partially adversarial: workload/network processes are exogenous and non-stationary, while trust and attack signal components depend on both the chosen action and the adversary's schedule. Therefore, we treat $\mathcal{M}$ as a POMDP in principle and operate the agent on the belief state s_t; Section 3.3.2 quantifies the effect of bounded adversarial corruption of observations.

Reward function R**.** The per-epoch reward balances performance, security, and stability:

$$R(s_t, a_t, s_{t+1}) = \omega_1 \widetilde{\mathrm{TPS}}_{t+1} - \omega_2 \tilde{L}_{t+1} - \omega_3 \Pi_{t+1} - \omega_4 \mathbb{1}[p_t \neq p_{t-1}]\, c_{\mathrm{sw}} \tag{6}$$

where $\widetilde{\mathrm{TPS}}$ and $\tilde{L}$ are normalized throughput and mean confirmation latency, $\Pi_{t+1} = \rho_1 n_{\mathrm{slash}} + \rho_2 \mathbb{1}[\text{safety alarm}] + \rho_3 \mathbb{1}[\text{liveness stall} > \tau_{\mathrm{stall}}]$ is the security penalty, and the final term imposes a switching cost c_{sw} whenever the protocol identity changes. The weights

satisfy $\omega_i > 0$ and are set so that a single safety violation outweighs any achievable per-epoch throughput gain ($\omega_3 \rho_2 > \omega_1$).

Objective. The controller seeks a policy $\pi_\theta : \mathcal{S} \to \Delta(\mathcal{A})$ maximizing the expected discounted return (Equation (7)):

$$J(\pi_\theta) = \mathbb{E}_{\pi_\theta, P}\Big[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t, s_{t+1})\Big], \qquad \gamma \in (0,1). \tag{7}$$

3.1.6. Design Goals as a Constrained Optimization Problem

Putting the pieces together, AdaTrust solves the constrained optimization of Equation (8):

$$\begin{aligned} \max_{\pi_\theta} J(\pi_\theta) \quad \text{s.t.} \quad & \Pr[\mathrm{Byz}(C_e) \geq \lceil k/3 \rceil] \leq \epsilon \ \forall e, \\ & \text{no conflicting finality during handovers, } N_{\mathrm{sw}}(T) \leq \sigma_{\max} T, \\ & \lim_{m \to \infty} \left| \mathbb{E}[T_i^{(m)}] - \theta_i \right| = 0 \ \forall i \end{aligned} \tag{8}$$

respectively corresponding to (G1) committee safety, (G2) switching safety, (G3) stability, and (G4) estimator consistency. The trust score must converge to each validator's true per-action honesty probability as evidence accumulates (the shrinkage bias of the Beta posterior vanishes as $O(1/m)$, Lemma 1) so that no validator is systematically advantaged or disadvantaged by the estimation procedure itself. Section 3.2 presents the mechanisms, while Section 3.3 maps each constraint to a theorem.

3.2. The AdaTrust Framework

3.2.1. Overall Architecture

AdaTrust is organized as two interacting control loops wrapped around an unmodified protocol pool (Figure 1).

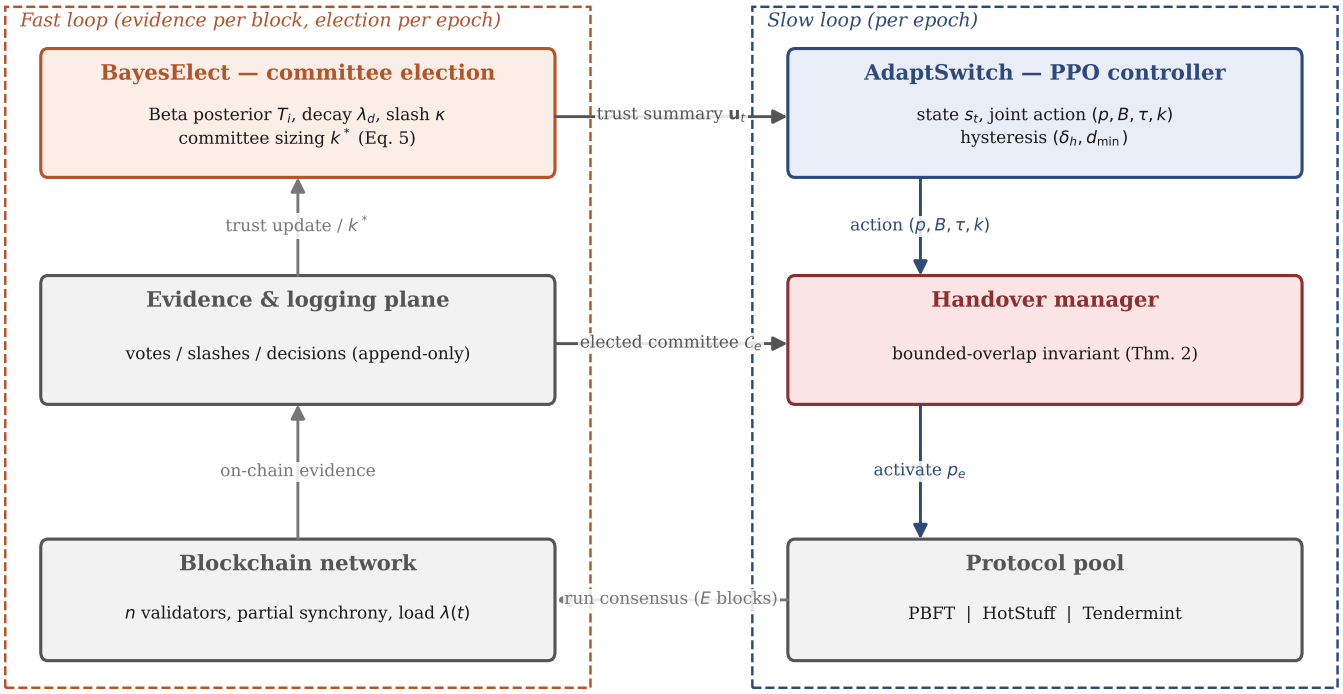


Figure 1. AdaTrust dual-loop architecture, showing the fast trust loop (BayesElect) that governs per-block committee membership and the slow DRL loop (AdaptSwitch) that governs the per-epoch protocol identity and configuration.

- **Fast loop (intra-epoch)—BayesElect.** At every epoch boundary, the trust layer ingests on-chain voting evidence, updates Beta posteriors (Equations (1) and (2)), computes the safety floor k^*, and elects the committee $\mathcal{C}_e$ by trust-weighted sampling.
- **Slow loop (cross-epoch)—AdaptSwitch.** A PPO agent observes the belief state s_t and selects the joint action $a_t = (p_t, B_t, \tau_t, k_t)$; a hysteresis filter admits the action only when the value margin and dwell time conditions hold.
- **Handover manager.** Enforces the bounded-overlap invariant, in which the outgoing protocol instance drains all pending heights before the incoming instance activates (Section 3.2.4).
- **Evidence and logging plane.** All votes, slashes, decisions, and performance samples are committed to an append-only log which feeds both the trust layer and the training pipeline, thereby guaranteeing reproducibility.

The loops operate at different timescales, which is by design: committee membership reacts within one epoch to deal with fast-moving misbehavior; as protocol identity represents a more expensive decision, it changes only based on sustained environmental shifts.

3.2.2. BayesElect: Bayesian Trust-Driven Adaptive Committee Election

Committee size. Let $\tilde{f}_t$ denote the effective adversarial mass under trust-weighted sampling: $\tilde{f}_t = \frac{\sum_{i\in\mathcal{B}_t} w_i}{\sum_{i\in\mathcal{N}} w_i}$ with $w_i = T_i^\eta$. Sampling k committee members yields a Byzantine count $X \sim \text{Binomial}(k, \tilde{f}_t)$ (without-replacement sampling is stochastically dominated; cf. Theorem 1). BayesElect selects the smallest committee meeting the safety floor:

$$k^* = \min\left\{k \in [k_{\min}, k_{\max}] : \Pr\left[X \geq \lceil k/3 \rceil\right] = \sum_{j=\lceil k/3 \rceil}^{k} \binom{k}{j} \tilde{f}_t^j (1-\tilde{f}_t)^{k-j} \leq \epsilon\right\}. \tag{9}$$

When the trust layer is informative ($\tilde{f}_t \ll 1/3$), k^* is far below static $3f+1$ sizing; when trust collapses, Equation (9) pushes k^* towards $k_{\max}$, gracefully degrading to conventional sizing. Since $\tilde{f}_t$ is not directly observable, it is estimated from the posterior: validators with T_i below the detection threshold θ_{det} contribute their weight to $\hat{f}_t$. For full transparency on the implemented tail, the deployed harness (Section 3.4.2) evaluates the binomial tail of Equation (9) at the runtime point estimate $\hat{f}_{\text{eff}} = \max(\hat{f}_t, n_{\text{det}}/n)$. It does not fold in the depletion factor c_w, which enters only the analytical bound of Theorem 1. The correction is a worst-case device; at the measured parameters, it would saturate the sizing at $k_{\max}$, whereas the point estimate keeps the rule operational. The floor $\hat{f}_{\text{eff}} \geq n_{\text{det}}/n$ guarantees reactivity to detections, and the gap between the analytical condition and the deployed rule is stated as well.

Complexity. Evidence extraction is $O(E \cdot k)$, posterior updates and weights are $O(n)$, the sizing loop evaluates at most $k_{\max} - k_{\min}$ binomial tails, each $O(k)$, i.e., $O(k_{\max}^2)$ in the worst case, and sampling is $O(n)$ with the alias method, for a total of $O(n + Ek)$ per epoch.

Trust decay and incentive alignment. Decay (Equation (2)) ensures that accumulated honesty cannot be "banked" indefinitely; maintaining selection weight requires ongoing correct behavior, thereby neutralizing on–off strategies (A2) in which the inactive periods erode T_i. Slashing (κ) makes equivocation (A1) immediately costly, as a single cryptographic proof outweighs κ epochs of good behavior.

3.2.3. AdaptSwitch: DRL-Driven Adaptive Protocol Switching

State, action, reward follow Equations (4)–(6); the agent acts at epoch granularity.

Why PPO? Three candidate families were considered. DQN-style methods are suitable for purely discrete actions but handle the joint space via a flat combinatorial action set, which scales poorly and destabilizes under the switching cost term. SAC targets continuous

control and maximum entropy exploration, which is an awkward fit for our categorical protocol head and has already been adopted in a concurrent work (DESAC [19]. Meanwhile, PPO [14] supports hybrid factorized policies with clipped stable updates, and is the established choice in recent consensus adaptation systems [18,20]. Here, we adopt PPO with a factorized policy (Equation (10)):

$$\pi_\theta(a_t \mid s_t) = \pi_\theta^p(p_t \mid s_t)\ \pi_\theta^B(B_t \mid s_t)\ \pi_\theta^\tau(\tau_t \mid s_t)\ \pi_\theta^k(k_t \mid s_t). \tag{10}$$

Network structure. A shared trunk of two fully connected layers (256 units each, tanh) feeds four categorical policy heads (protocol: three-way; block size, timeout, committee hint: four-way each) and a scalar value head (≈150k parameters).

Hysteresis mechanism. AdaptSwitch switches only when both conditions hold (Equation (11)):

$$\hat{VI}^\pi(s_t, a_t^*) - \hat{VI}^\pi(s_t, a_t^{\text{cur}}) > \delta_h, \qquad t - t_{\text{last_sw}} \geq d_{\min} \tag{11}$$

where $a_t^* = \arg\max_a \hat{Q}(s_t, a)$ and a_t^{cur} continues the current protocol with refreshed parameters. Theorem 4 (Section 3.3.2) bounds the switching frequency and the hysteresis-induced suboptimality.

We provide the following implementation note covering how the action-dependent values are obtained. No action-conditioned critic is maintained; the candidate a_t^* is obtained by deterministic argmax over the factorized policy heads π_θ of Equation (10) (one argmax per head), and the values $\hat{VI}^\pi$ in Equation (11) are operationalized through the shared scalar value head $V_\phi(s_t)$. Because the state s_t (Equation (4)) already encodes the operating regime consisting of the trust landscape, load statistics, and one-hot encoding of the active protocol, the magnitude of the value estimate serves as a confidence proxy for the comparison in Equation (11) and the gate admits the candidate protocol component only when $|V_\phi(s_t)| > \delta_h$ together with the dwell condition. Both the training rollouts (Algorithm 1) and the deployed controller use this single value head; $\hat{Q}$ in Algorithm 1 denotes this value head evaluation of the candidate state under the argmax action.

Algorithm 1 AdaptSwitch Training (PPO with hysteresis-aware rollout)

```
Input: simulator env E(.); policy/value nets theta, phi; clip epsilon_c; GAE lambda_g; discount gamma;
       hysteresis (delta_h, d_min); switching cost c_sw; epochs per iter H; iterations M
Output: trained policy pi_theta
1: for iter = 1 ... M do
2:      D <- empty; s <- env.reset(); p_cur <- PBFT; t_last_sw <- -inf
3:      for t = 1 ... H do
4:          a ~ pi_theta(. | s); a* <- argmax over pi_theta heads; v <- V_phi(s)
5:          a_cur <- (p_cur, a.B, a.tau, a.k)              (current action, Equation (11))
6:          if a*.p != p_cur and |v| > delta_h and t - t_last_sw >= d_min then
7:              p_cur <- a*.p; t_last_sw <- t
8:          a_exec <- (p_cur, a.B, a.tau, a.k)
9:          s', r <- env.step(a_exec)                       (r from Equation (6))
10:         D <- D union {(s, a_exec, r, s')}; s <- s'
11:     A^ <- GAE(D, V_phi, gamma, lambda_g)
12:     for epoch = 1 ... 10 do
13:         theta <- argmax_theta sum_D min(r_theta A^, clip(r_theta, 1-epsilon_c, 1+epsilon_c) A^)
14:         phi <- argmin_phi sum_D (V_phi(s) - R^)^2
15: return pi_theta
```

Complexity. Rollout cost per epoch is dominated by the simulator's consensus execution; the PPO update is $O(|D| \cdot |\theta|)$ on GPU and is offline. Online inference is $O(|s| \cdot 256)$ FLOPs—three orders of magnitude below one consensus round at $k \geq 4$ (Section 3.3.3).

3.2.4. Consensus Flow Integration

An AdaTrust epoch proceeds in five phases, shown in Figure 2. (1) Observe: The evidence plane finalizes the epoch log; trust updates run (Algorithm 2, lines 1–7). (2) Decide:

AdaptSwitch evaluates the hysteresis gate; the result is (p_e, B_e, τ_e) and the committee hint k_e. (3) Elect: BayesElect computes $k^* \vee k_e$ and samples C_e; membership is committed to the log. (4) Handover: The outgoing instance finalizes every height up to h^* (drain); the incoming instance activates at $h^* + 1$. Bounded-overlap invariant: At every blockchain height, exactly one protocol instance is empowered to decide. (5) Execute: The elected committee runs p_e for E blocks; votes stream into the evidence plane. Safety of phase 4 is proved as Theorem 2; liveness follows from per-protocol liveness plus finite drain time under partial synchrony.

Algorithm 2 BayesElect (per epoch e)

```
Input: validator set N; posteriors {(alpha_i, beta_i)}; evidence log L_{e-1};
       sharpness eta; decay lambda_d; slash kappa; tolerance epsilon; bounds (k_min, k_max)
Output: committee C_e; updated posteriors {(alpha'_i, beta'_i)}
1: for each i in N do
2:      (s_i, m_i, q_i) <- ExtractEvidence(L_{e-1}, i)
3:      alpha_i <- alpha_i + s_i; beta_i <- beta_i + m_i + kappa.q_i
4:      alpha_i <- 1 + lambda_d(alpha_i - 1); beta_i <- 1 + lambda_d(beta_i - 1)      (Equation (2))
5:      T_i <- alpha_i / (alpha_i + beta_i)                                  (Equation (1))
6:      w_i <- T_i^eta
7: f^ <- sum_{i : T_i < theta_det} w_i / sum_{i in N} w_i
8: k* <- smallest k in [k_min, k_max] satisfying Equation (9) with f^
9: C_e <- WeightedSampleWithoutReplacement(N, w, k*)
10: return C_e, {(alpha_i, beta_i)}
```

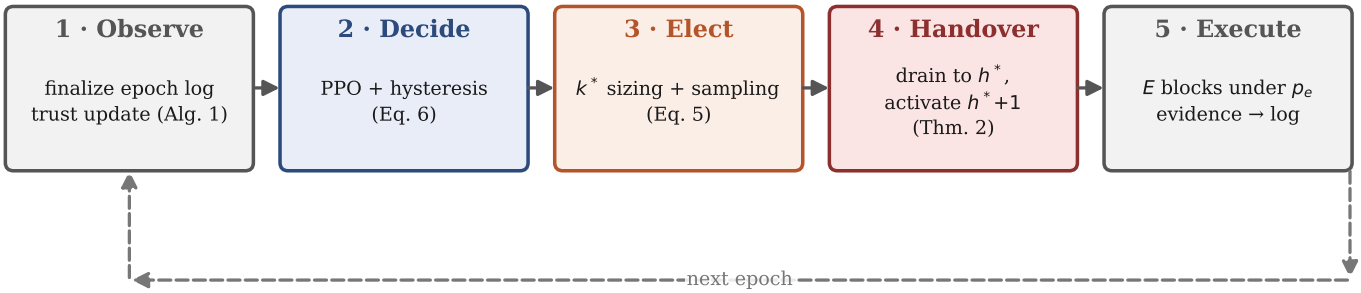


Figure 2. Epoch flow with the bounded-overlap handover invariant: observe, decide, elect, drain-and-handover, execute.

This five-phase flow is the design model. Section 3.4.2 (the deployment mapping table) describes how each phase is realized on the real-chain testbed; in particular, phase 4 is realized as chain-level drain-and-reconnect rather than an in-place swap, so Theorem 2's invariant is not instantiated there (see the implementation status remark after Theorem 2). With respect to state transfer across a handover, the handover transfers responsibility for heights, not the ledger state. The outgoing instance drains every pending height to finality under its own quorum (phase 4), so the committed application state remains owned and finalized by the outgoing chain. Each height is decided by exactly one instance, and only then is the load stream rerouted to the incoming chain, which proposes exclusively at fresh heights. Pending transactions that the outgoing chain cannot finalize before the switch are re-issued by the harness after the drain; in our evaluation, these are load-generated transfers with no cross-chain dependencies, so no state carry-over is required. In a single-ledger in-place deployment, the same drain-then-activate sequence runs within one ledger, and no cross-chain state question arises at all; in this way, the dual-chain testbed is a conservative performance-oriented realization of the handover. End-to-end instantiation of the invariant remains as future work (Section 5.4).

3.2.5. Compatibility with Existing Protocols

AdaTrust imposes three requirements on a candidate protocol: (i) a parameter interface exposing at least block size and timeout; (ii) deterministic committee injection (the protocol must accept an externally supplied validator set per epoch, which is also true

of PBFT, HotStuff, and Tendermint-style designs); (iii) a decision log from which vote evidence can be extracted; and, by the threat model, (iv) Byzantine fault tolerance, with CFT protocols inadmissible to the pool (Section 3.1.2). No message format or cryptographic changes are needed, keeping AdaTrust deployable as an orchestration layer above stock implementations.

3.3. Theoretical Analysis

3.3.1. Security Analysis

Assumption 1. *(A1) Cryptographic primitives are secure. (A2) Each protocol $p \in \mathbb{P}$ is individually safe under its own fault assumption [2,3]. (A3) On-chain vote evidence is objective. (A4) After a warm-up of m_0 epochs, the adversary's per-epoch trust weight is bounded by the posterior estimate with high probability (Lemma 1).*

Lemma 1 (**Posterior concentration**)**.** *For a validator with true per-action honesty probability θ_i, the posterior mean after m evidence actions satisfies $\Pr[|T_i - \theta_i| > \xi] \leq 2\exp(-2m\xi^2)$ (before decay).*

Proof. The evidence is m i.i.d. Bernoulli(θ_i) outcomes (A3); the posterior mean is a shrinkage estimator with bias $\leq (\alpha_0 + \beta_0)/m$. Hoeffding's inequality on the sample mean, with the bias absorbed into ξ, yields the claim. □

Theorem 1 (**Committee corruption bound**)**.** *Under A1–A4, let $\tilde{f}_t$ denote the depletion-corrected effective adversarial mass under trust-weighted without-replacement sampling, i.e., $\tilde{f}_t := \min\{1, c_w \hat{f}_t^{\text{raw}}\}$, with the weight-depletion factor $c_w = \frac{n}{n-k}\frac{w_{\max}}{\bar{w}}$ defined in the proof, and assume $\tilde{f}_t < 1/3$. With k chosen by Equation (9), the elected committee contains at least $\lceil k/3 \rceil$ Byzantine validators with probability at most ϵ, and*

$$\Pr[\text{corrupt}(C_e)] \leq \exp\left(-2k\left(\tfrac{1}{3} - \tilde{f}_t\right)^2\right) \tag{12}$$

Proof. By a weighted-sampling coupling, if at every without-replacement draw the conditional adversarial probability is at most q, then the adversarial count is stochastically dominated by Binomial(k, q). Under A4, the raw aggregate adversarial fraction is $\hat{f}_t^{\text{raw}} < 1/3$; with per-validator weight clipped at $w_{\max}$ (Equation (2)), weight depletion inflates the conditional adversarial fraction of later draws by at most $c_w = \frac{n}{n-k}\frac{w_{\max}}{\bar{w}}$, so every draw faces an adversarial probability of at most $\tilde{f}_t = \min\{1, c_w \hat{f}_t^{\text{raw}}\}$. Requiring $\tilde{f}_t < 1/3$ and applying Hoeffding at threshold $k/3$ yields Equation (12); Equation (9) selects the smallest k for which the exact binomial tail at the same depletion-corrected mass is within ϵ. □

Remark 1 (Consistency of Equation (9), (12) and A4)**.** *The depletion correction enters the bound through a single quantity $\tilde{f}_t$. Equation (12) is evaluated at $\tilde{f}_t$. The deployed sizing rule (Section 3.2.2) evaluates the tail of Equation (9) at the point estimate $\hat{f}_{\text{eff}}$ without the c_w factor. A4 holds by the estimation chain of Section 3.1.3. Vote evidence is objective (A3); by Lemma 1, the posterior mean concentrates around the true honesty probability at rate $2\exp(-2m\xi^2)$, so after the warm-up of m_0 epochs the estimated adversarial mass deviates from its true value by at most an $O(\xi)$-scale error with high probability. In addition, the decay of Equation (2) prevents trust from being banked across silent periods. The residual estimation error is absorbed into the safety margin of the k-selection step. The deployed value of $\epsilon = 10^{-3}$ leaves orders of magnitude of headroom over the $O(\xi)$-scale error at the measured evidence volumes.*

Remark 2 (Manipulation resistance)**.** *Raising selection weight requires accumulating posterior mass, which is useful consensus work (A3). Decay prevents banking through idle periods, and a*

single equivocation erases $\Theta(\kappa)$ epochs of accumulation, so $\tilde{f}_t$ never exceeds the raw Byzantine ratio in expectation.

Definition 1 (Per-protocol finality and drain completion)**.** *A block is final in instance $I = (p, \mathbb{C})$ under PBFT/Tendermint's commit certificate ($\geq \lceil 2|\mathbb{C}|/3 \rceil$ votes, $\delta_p^{\text{fin}} = 1$) or HotStuff's three-chain of quorum certificates ($\delta_p^{\text{fin}} = 3$). A drain of I completes at the first height h_d where every block proposed in I is final. The bounded-overlap invariant is that the successor instance proposes only at heights $> h_d$, and one instance at most is active at every height.*

Theorem 2 (Switching safety)**.** *Under A1–A2 and the bounded-overlap invariant with per-protocol drain completion, provided that each active committee satisfies Theorem 1, AdaTrust never finalizes two conflicting blocks at the same height across any sequence of switches.*

Proof. Two conflicting final blocks at height h would be witnessed by the unique active instance I (the invariant assigns each height to one instance). For PBFT/Tendermint, two distinct commit certificates at (h, view) intersect in $\geq \lceil |\mathbb{C}|/3 \rceil$ validators, at least one of which is honest, contradicting one-commit-per-height. For HotStuff, conflicting three-chains imply an honest double-voter. By Theorem 1 the committee meets its corruption bound except with probability $\leq \epsilon$, contradicting A2. Union-bounding over E_{tot} epochs gives $\Pr[\text{any safety failure}] \leq E_{\text{tot}}\epsilon$; holding system risk at ϵ_{sys} requires $\epsilon = \epsilon_{\text{sys}}/E_{\text{tot}}$, costing only $O(\log E_{\text{tot}})$ seats. □

Remark 3. *Theorem 2 is a design-time guarantee. Our testbed runs the two backends as independent chains (no in-place swap), so the invariant is not exercised end-to-end (Section 3.4); testbed instantiation is future work (Section 5.4).*

Proposition 1 (**Drain liveness**)**.** *After GST, with a committee satisfying Theorem 1 and timeout τ at least the maximum message delay, a drain completes within $\delta_p^{\text{fin}} + 2$ rounds (at most 5τ); pre-GST, it may be delayed, but by Theorem 2 is never unsafe.*

Proof Sketch. After GST, each $p \in \mathcal{P}$ is live: a correct leader drives a decision within one τ-bounded round-trip, with at most two view changes; finality then needs δ_p^{fin} further rounds. Summing gives the bound. □

3.3.2. Convergence Analysis

Assumption 2. *(B1) Within each regime, the induced MDP is stationary and ergodic. (B2) Rewards are bounded: $|R| \leq R_{\max}$. (B3) Step sizes satisfy Robbins–Monro conditions. (B4) At most a fraction $\rho < 1$ of the reward per epoch is adversarially steerable.*

Theorem 3 (**Policy convergence, conditional**)**.** *If B1–B3 hold, PPO converges to a stationary point of $J(\theta)$; for tabular-softmax policies, the limit is a local optimum. This is conditional, as B1 is an environment property that our workload only approximates, so the empirical burden is carried by the real-chain evaluation (Section 4).*

Proof Sketch. The clipped surrogate lower-bounds improvement within a trust region [14]; under B1–B3, the iteration satisfies non-convex stochastic approximation conditions, giving $\nabla J(\theta_t) \to 0$ almost surely. □

Theorem 4 (**Hysteresis-induced suboptimality and switching bound**)**.** *Let π^* be the optimal unconstrained policy; let π_h be the hysteresis-gated policy with margin δ_h and dwell $d_{\min}$. Then: (i) $N_{sw}(T) \leq \min\{T/d_{\min},\ J(\pi^*)/(\omega_4 c_{sw})\}$, where the second branch additionally requires $J(\pi^*) > 0$,*

i.e., the optimal policy earns positive expected discounted return, which holds under the deployed reward normalization, where the normalized throughput term is non-negative and dominant so that even the static no-switch policy earns $J > 0$ and π^* can only improve on it; and (ii)

$$J(\pi^*) - J(\pi_h) \leq \frac{\delta_h}{1-\gamma}. \tag{13}$$

Proof Sketch. (i) The dwell caps the switch rate at $1/d_{\min}$. For the second branch, any policy executing N switches pays at least $N \cdot \omega_4 c_{\text{sw}}$ in switching costs within its total return; since no policy's return exceeds $J(\pi^*)$, $N_{\text{sw}} \leq J(\pi^*)/(\omega_4 c_{\text{sw}})$ whenever $J(\pi^*) > 0$ (the condition stated above). Here, (ii) is stated for the abstract action value margin form of Equation (11); the deployed confidence proxy (see Section 3.2.3, implementation note) directly preserves the dwell-based bound in (i) and shares the parameter δ_h, while its suboptimality behavior is measured empirically by the hysteresis ablation (see Section 4.4; removing the gate inflates run-to-run variance 3.4-fold without a significant mean shift). (ii) The gate blocks a switch only when the value margin is at most δ_h; the blocked switch saves $\omega_4 c_{\text{sw}}$, so the gap is bounded by the gated margin propagated through the discount factor, yielding Equation (13). □

Theorem 5 (**Robustness to adversarial reward corruption**). *Under B1–B4, let $\hat{\pi}$ be optimal in the reward-perturbed MDP and let π^* be optimal in the true MDP; then, $J(\pi^*) - J(\hat{\pi}) \leq \frac{2\rho R_{\max}}{1-\gamma}$.*

Proof Sketch. The per-step perturbation is bounded by $\varepsilon_r = \rho R_{\max}$ (B4); the simulation lemma gives $\|VIII^{\pi}_{\text{true}} - VIII^{\pi}_{\text{pert}}\|_\infty \leq \varepsilon_r/(1-\gamma)$, and the policy loss is at most twice the value gap.

Interpretation. The trust layer keeps ρ small; each slashing event caps the adversary's steering power. □

3.3.3. Communication and Computation Complexity

Table 2 shows the per-round message complexity for committee size k among n validators.

Table 2. Per-round message complexity.

Protocol	Normal Case	View Change	Finality
HotStuff	$O(k)$ (threshold sig.)	$O(k)$	three-chain
Tendermint	$O(k \log k)$ (gossip)	$O(k \log k)$	immediate
PBFT	$O(k^2)$	$O(k^2)$	immediate

Trust updating costs $O(n)$ per block, while BayesElect costs $O(n + k^2_{\max})$ per epoch. The deployed controller (2×256 tanh trunk, four categorical heads + value head, ≈150 k parameters) costs $\approx 3 \times 10^5$ FLOPs per forward pass, which is one to two orders of magnitude below the per-round signature budget at $k = 4$ (32 Ed25519 verifications ≈1.6–3.2 ms, measured) and constant in n. Since Theorem 1 admits $k = O(\log(1/\epsilon))$ at fixed $\tilde{f} < 1/3$, effective message complexity is $O(\log^2(1/\epsilon))$ in PBFT mode versus $O(n^2)$ for full-set PBFT; this is the formal basis of the committee overhead results in Section 4.2.

3.3.4. Theoretical Comparison with Existing Protocols

Table 3 contrasts AdaTrust with representative protocols across message complexity, adaptation scope, and the guarantees that each provides.

Table 3. Theoretical comparison. N/A: not applicable; ×: not supported; ✓: supported. The row in bold highlights the proposed system (AdaTrust).

Scheme	Message/Round	Adaptation	Committee Bound	Switching Safety	Policy Convergence
PBFT [2]	$O(n^2)$	none	N/A (static n)	N/A	N/A
HotStuff [3]	$O(n)$	none	N/A (static n)	N/A	N/A
DA-PBFT [17]	$O(n^2)$	parameters only	×	× (no switches)	empirical only
DAGWise [18]	$O(n)$	parameters only	×	×	empirical only
rPoA [22]	$O(k^2)$	membership (heuristic)	× (no sizing rule)	×	N/A
AdaTrust	$O(k^{*2})$, $k^* = O(\log \frac{1}{\epsilon})$	**membership + protocol + parameters**	✓ **(Thm. 1)**	✓ **(Thm. 2)**	✓ **(Thm. 3–5)**

Assumption register: A1–A4 (security) and B1–B4 (convergence) are the complete set; violations degrade gracefully. A4 failure reduces to static $3f + 1$ sizing via $k_{\max}$, and B4 violation ($\rho \to 1$) voids Theorem 5 while leaving Theorems 1 and 2 unaffected.

3.4. Experimental Setup

3.4.1. Real-Chain Testbed

All results in Section 4 are measured on live blockchain deployments; the only exception is the w/o-switching-cost ablation in Section 4.4, which is a training-time variant evaluated offline in the training environment (four seeds) and is reported as evidence about training dynamics only. No other number in Section 4 is produced by simulation. The testbed runs on a shared GPU node (2× NVIDIA H100 80GB, 251 GB RAM, Ubuntu 22.04) and comprises two production-grade stacks: a 16-validator CometBFT v0.38.17 network (one Docker container per validator; committee reconfiguration actuated at runtime through ABCI `ValidatorUpdate` power changes) and a four-validator Hyperledger Besu 24.12.2 QBFT network (one JVM per validator) that provides the fixed small committee BFT reference. An experiment harness (Python 3.10) ingests every committed block via RPC, updates the Beta trust layer per block, evaluates the trained PPO policy in argmax mode, issues committee updates, and drives load through a wall-clock-paced transaction blaster; all actuation is by real transactions and real container control (`docker stop/start`). Control decisions are taken at harness epochs of 13.7–16.6 s (36–43 epochs per 600 s run). Throughput, campaigns run two independent 16-validator CometBFT networks (chains A/B, cloned genesis) plus the Besu network; with both streams active the 32 cores are oversubscribed (load ≈ 38), so cross-stream absolutes are not comparable. Every paired comparison is constructed within one stream by interleaved round-robin (Section 3.4.4).

3.4.2. Sim-to-Real Methodology and Hyperparameters

The discrete-event simulator (Section 3.1.1) provides the training environment for the PPO controller. Its parameters were calibrated against the real CometBFT deployment before training, after which the policy was frozen and deployed unchanged (argmax inference, no exploration, no online updates). The training hyperparameters are summarized in Table 4.

Deployment mapping. Protocol 0 (PBFT) → Besu QBFT; protocol 1 (HotStuff) → remapped to CometBFT (no HotStuff implementation exists; Tendermint is its nearest operational niche); protocol 2 (Tendermint) → CometBFT; committee hints 4, 8, 16, 32 clamp to $[4, 16]$ via ABCI `ValidatorUpdate`; block-size and timeout heads are **not actuated on-chain** (neither backend exposes hot parameter updates) and only influence training. Consequently

the deployable protocol set has cardinality two, and committee sizing plus protocol identity are the only live control channels.

Simulator fidelity. The calibrated simulator lands within MAPE 4.6% of measured commit latency across netem-injected delay tiers (0–100 ms, 60 samples/tier), within +4.0% on Tendermint-niche service rate (680 vs. 653.7 TPS, Table 5), and ≈±20% on the PBFT niche (vs. QBFT). Two mismatches survive calibration: (i) QBFT's real bottleneck is per-sender admission throttling (Section 4.1), a mempool policy with no simulator counterpart; and (ii) the simulator has no proposer-quality heterogeneity, a real-chain effect to which small committees are exposed (Section 5.4), so training never sees it. Both push the policy toward overvaluing niches the real pool does not have (RQ4's `noswcost` collapse); this is why every performance claim in Section 4 (except for the ablation variant in Section 4.4 without the switching cost term) is measured on the live chains, with the simulator treated as a training environment rather than evidence.

Table 4. PPO training hyperparameters.

Component	Setting
PPO	lr 3×10^{-4}, $\gamma = 0.99$, clip 0.2, GAE 0.95, 10 epochs, batch 2048, entropy 0.01; 400–500 iters × 256-epoch rollouts (≈480–580 s/seed on one H100)
Network	shared 2×256 tanh trunk; 4 categorical heads (3/4/4/4) + value head
Reward	$\omega_1 = 3.0$, $\omega_2 = \omega_3 = \omega_4 = 1.0$
Hysteresis/trust	$\delta_h = 0.05$, $d_{\min} = 5$ epochs; $\lambda_d = 0.98$, $\kappa = 50$, $\eta = 2$, $\theta_{\text{det}} = 0.35$, $k \in [4, 16]$, $\epsilon = 10^{-3}$; $\hat{f}_{\text{eff}} = \max(\hat{f},\ n_{\text{det}}/16)$
Statistics	interleaved paired runs; $n = 6$ pairs (RQ1, RQ3, RQ4, RQ5); paired t-tests with Holm correction, Cohen's d_z

Table 5. RQ1 real-chain results (n=6 paired rounds; per-round paired t-tests vs. AdaTrust, df = 5; $d_z = \overline{\Delta}/\text{SD}(\Delta)$; 95% CI of the paired difference in brackets). Bold marks the AdaTrust row.

Scheme	TPS (Mean ± std)	Switches	Drain/Switch (s)	Mean Backlog	t	p	Cohen's d_z	95
AdaTrust (TM, $k = 4$)	**653.7 ± 4.5**	0	—	806 ± 29	—	—	—	—
B2 StaticTM ($k = 16$)	659.2 ± 9.2	0	—	916 ± 72	−1.10	0.32	−0.45	[−18.2, +7.3]
B1 StaticQBFT	84.3 ± 1.5	0	—	830 ± 30	267.2	$\approx 10^{-11}$	109.1	[+563.9, +574.9]
B3 Reactive switcher	224.4 ± 20.0	15.0	10.8 ± 9.4	566 ± 39	42.9	$\approx 10^{-7}$	17.5	[+403.6, +455.0]
B4 Parameter-only PPO	648.4 ± 6.2	0	—	—	−0.45	≈0.69	−0.26	[−38.0, +30.8]

3.4.3. Baselines

- **B1—Static-QBFT**: The Besu 4-validator QBFT network, the fixed-small-committee reference (a matched-workload operating point, not QBFT's peak; see Section 4.1).
- **B2—StaticTM** [38]: Full 16-validator CometBFT, no trust layer, $k = 16$ throughout.
- **B3—Reactive threshold switcher**: A rule-based switcher moving load to CometBFT ($k = 16$) above 600 tx/s and to QBFT ($k = 4$) otherwise, with minimum-dwell debounce. The real measured drain is paid out at every handover, providing a realistic "adaptive" baseline without trust or learning.
- **Clairvoyant oracle (degenerate).** Here, the per-phase-best schedule degenerates to never switching (CometBFT dominates QBFT in both phases, see Table 5); in order to price reactive switching, we report B3 instead.
- **B4—Parameter-only PPO**: A controller trained with the protocol head pinned to Tendermint, isolating the protocol identity lever (with the trust layer active, as in AdaTrust).
- **Ablations (RQ4)**: Variants consist of w/o trust observations, w/o sizing (k frozen at 4), w/o hysteresis ($\delta_h = 0$, $d_{\min} = 0$), and w/o the switching cost term (ω_4=0, retrained, offline).

3.4.4. Metrics and Statistical Protocol

- **Throughput**: Committed tx/wall-clock seconds, cross-checked against chain height and block contents.
- **Retention**: Post-attack/pre-attack TPS per run (RQ3).
- **Detection latency**: Wall time from injection to the first block with $n_{\text{det}} \geq 1$ (validators with trust $< \theta_{\text{det}}$).
- **Healing/recovery:** Committee expansion k after detection; latency to $\geq 80\%$ of pre-attack moving-average TPS.
- **Reconfiguration cost**: Wall time and block count from `ValidatorUpdate` to on-chain effect (RQ2).
- **Statistics**: Schemes alternate run-by-run within one window (interleaved round-robin) while sharing machine load, chain state age, and blaster conditions; paired t-tests with Holm correction and Cohen's d_z. Absolute throughput drifts 3–6% across campaigns; thus, we only compare within-campaign paired differences, never absolutes across campaigns.

3.4.5. Workload Design

A wall-clock-locked square wave alternates 150 and 1200 tx/s with period $P \in \{60, 120, 240\}$ s ($P = 120$ s baseline). The high phase offers ≈1.8× CometBFT's measured capacity (≈650 TPS), stressing admission control and backlog; the low phase lets queues drain.

3.4.6. Attack Injection

Silence attack (RQ3). At $t = 300$ s, validator containers are stopped (`docker stop`), the canonical pragmatic failure mode. The dose is calibrated per mode towards the maximal silent set that preserves liveness: AdaTrust loses 1 of its $k = 4$ committee (25%), while StaticTM loses 3 of 16 voters (18.75%). The quorum arithmetic bounds this; CometBFT finalizes with >2/3 of 16 votes (≥11), so StaticTM tolerates at most five silent validators (31.25%) and halts only at the sixth. The 3/16 dose is deliberately lenient, and a five-victim re-run (Section 4.3) shows that the retention gap persists (0.808 vs. 0.909). Evidence comes from `last_commit` signatures; one subtlety proved load-bearing (Section 4.3): CometBFT v0.38 returns ABSENT `last_commit` entries with an empty validator address, so evidence must be mapped positionally against the block's `validators_hash` or the silence is invisible.

Coverage statement. The threat model (Section 3.1.2) spans equivocation, strategic silence, collusion, and switching-aware adaptation; the live evaluation instantiates crash silence only, which is the one class injectable without forking consensus implementations. The rest are treated analytically (Section 3.3); on-chain double-sign injection is future work (Section 5.4).

3.4.7. Experiment Groups

RQ1 headline throughput (4 schemes × 6 paired rounds × 600 s); RQ2 (a) reconfiguration cost (batch $B \in \{1, 4, 12\}$ × 3 reps) and (b) committee-capacity scan ($k \in \{4, 8, 12, 16\}$); RQ3 silence-attack resilience (2 modes × 6 runs × 600 s); RQ4 ablations (4 variants × 6 runs); RQ5 halt-regime escape (2 modes × 6 paired runs × 600 s; 6 of 16 validators stopped, below the $\lceil 2n/3 \rceil = 11$ quorum). Raw epoch-level records are exported as CSV under `runs/real/rq1..5/`; figures regenerate at 300 DPI in PNG and PDF [49].

4. Results

4.1. RQ1: Headline Throughput vs. Baselines (Table 5, Figure 3)

Configuration. Square-wave workload ($P = 120$ s, 150/1200 tx/s), six paired rounds of 600 s per scheme, interleaved round-robin on chain A; AdaTrust deploys the frozen argmax policy with the trust layer active.

Learned control (Section 3.4.3), $n = 3$ paired rounds in a separate campaign window; $t/p/d_z$/CI from the three index-paired differences against AdaTrust runs collected in the same window (AdaTrust 644.8 ± 8.0 TPS there). B4's committee head choices ($k = 8$) are actuated identically to AdaTrust's, while its protocol head is pinned to Tendermint in both training and deployment.

Statistical unit note. All tests in this table use the six paired rounds as the analysis unit. An earlier analysis used 30 s phase-matched buckets ($n \approx 120$); because buckets within a round are autocorrelated, that unit overstated the effective sample size, and we report it only in the artifact appendix with the nesting caveat. Per-round per-scheme values are released with the artifact.

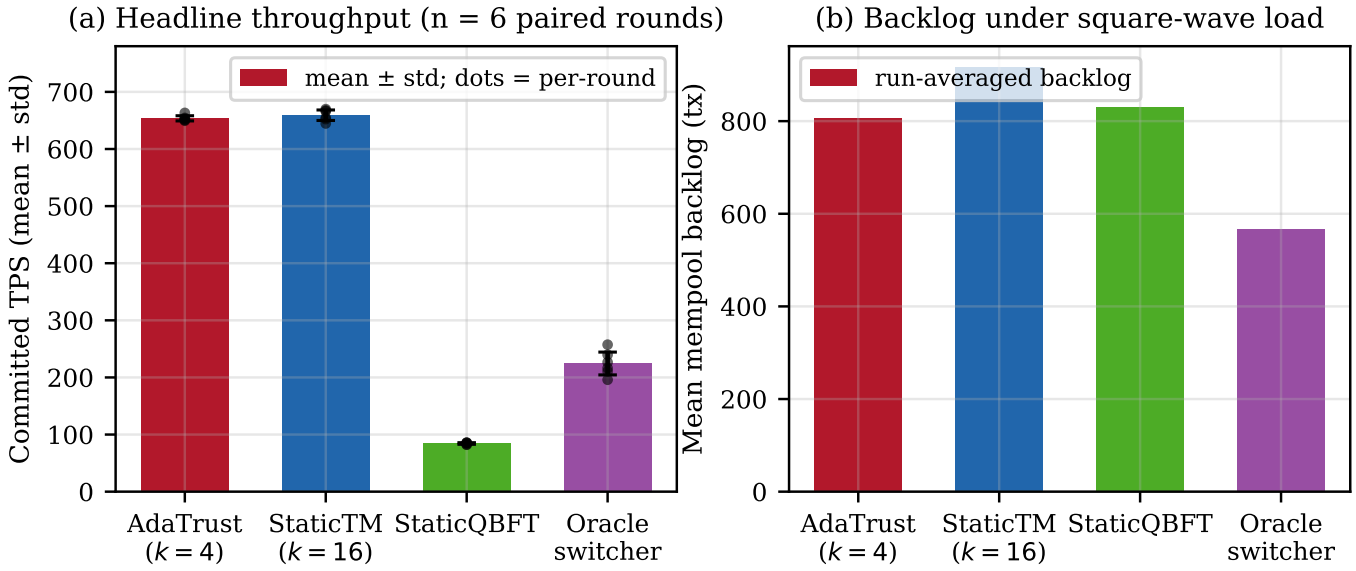


Figure 3. RQ1 headline throughput on the live chains: (**a**) per-scheme mean TPS ($n = 6$ paired rounds, interleaved on chain A); (**b**) backlog under the square-wave load.

Results. AdaTrust matches the full-16-validator StaticTM baseline (653.7 vs. 659.2 TPS, paired $p = 0.32$, 95% CI of the difference $[-18.2, +7.3]$, statistically indistinguishable) while voting with only $k = 4$. Against the four-node Besu QBFT deployment, it serves 7.8× the throughput (84.3 TPS, $d_z = 109$), while against the reactive threshold switcher it serves 2.9× the throughput (224.4 TPS, $d_z = 17.5$). Two scoping conditions accompany the QBFT comparison (Section 3.4.1): both stacks shared the oversubscribed machine, and the load generator used a fixed account set that trips Besu's per-sender admission cap. Therefore, the 84.3 TPS figure is a matched-workload operating point, not the protocol's ceiling. A dedicated fair-rig campaign (500 pre-funded senders, relaxed pool limits, machine otherwise idle) bounds that ceiling from below; the same 4-node network cleanly absorbs 394.2 of 400 offered TPS (98.5%) and sustains ≈462 TPS at 600 offered (≈461-tx blocks at 1 s periods, pool depth bounded ≈315), with our injector's per-process signing cap (105 TPS per shard, eight shards) halting the probe before the chain does. Thus, the 7.8× headline is a ratio between deployed operating points, not protocol capabilities; QBFT's fair-baseline throughput is ≥5.5× the matched-workload figure. The deployed policy's behavior is itself a result: across all six rounds and both load phases, it never switches protocol, and after the initial election (epoch 1) it never changes the committee (0 drains, $k = 4$ throughout). The learned control B4 closes the loop on this null: with the protocol lever removed at training time, the parameter-only policy reaches the same throughput (648.4 vs. 644.8 TPS in-window, paired $p \approx 0.69$), and $k = 8$ where AdaTrust chooses $k = 4$, which is operationally equivalent under RQ2(b)'s flatness. This is exactly what the degenerate-oracle

analysis predicts: on a pool where one backend dominates both phases, protocol identity carries no exploitable value and a controller denied that lever loses nothing. The lever's value is conditional on the pool (structurally distinct niches), not intrinsic; what remains unconditional is the training stability role of the switching-cost term (Section 4.4) and the attack response value of the trust layer (Section 4.3), which B4 shares and is not what this control varies.

Analysis of where the simulator and the real chain part ways. Two findings rewrite the narrative we carried in from simulation. (i) The QBFT/PBFT niche is structural, not parametric: Besu's transaction pool bounds the number of queued future transactions per sender (`tx-pool-max-future-by-sender`, default 200); with a fixed account set generating load, this per-sender admission cap throttles the steady-state inflow that the pool will accept, and QBFT saturates at ≈84 TPS regardless of how the controller tunes block parameters. The binding constraint is admission, not block space; we observed single blocks physically committing over 1700 simple transfers (1750 txs at height 16,439, ≈36.75 M gas; artifact `runs/real/inj/inj_besu_1788823548.csv`, Appendix A), so the ceiling cannot be attributed to the block gas limit. (ii) Switching never pays on this pool, and the correct reference point is no-switch: B3's 65.7% loss decomposes into three measured components (Section 3.4.4), with TM legs covering ≈19% of wall time and serving at 633 TPS, near the static reference. QBFT legs cover ≈46%, but deliver only ≈20 TPS effective, as each ≈60 s trough leg is consumed by fresh-JVM and blaster/tx-pool warm-up and never reaches the 84.3 TPS steady state. Drain-containing switch epochs cover the remaining ≈35%, serving at ≈205 TPS. The three components weight to ≈224 TPS, reproducing the measured value; a naive 58% at 84.3 plus 27% at zero account does not do this, since QBFT legs never see 84.3 and drain epochs are not zero-service. Note that for the clairvoyant baseline, this implies that because CometBFT dominates QBFT in both load phases, the per-phase-best oracle schedule degenerates to never switching. The true upper bound on this pool is 653.7 TPS with zero switches, which is exactly the policy learned by AdaTrust. The 2.9× figure does not mean AdaTrust beats an oracle; rather, it means that a controller with reactive switching capability pays a large measured penalty. AdaTrust's value on this pool lies in recognizing that the switching lever has negative expected value here. B3's lowest backlog (566 vs. 806) is the visible signature of this penalty; switching drains the queue, but destroys service capacity. The learned argmax policy trained with the switching cost term and the hysteresis gate stays on CometBFT at $k = 4$; RQ4 examines what the training-time ablations say about how robust this refusal is. The takeaway is that on the real pool, the value of adaptation lies in trust-driven committee sizing and knowing when not to switch; the switching capability remains the insurance policy for deployments where the pool contains protocols with genuinely complementary structural niches (see Section 5.1).

4.2. RQ2: Reconfiguration Cost and Committee Capacity (Table 6, Figure 4)

(a) Reconfiguration cost. We submit batches of $B \in \{1, 4, 12\}$ `ValidatorUpdate` transactions (shrink to power 0; restore to power 1) under load and measure wall time and blocks to on-chain effect (three repetitions; ranges reported).

Table 6. RQ2 real-chain results: committee reconfiguration cost (three repetitions; ranges reported).

Batch B	Wall Time (s)	Blocks	Amortized Per-tx (s)
1	2.4–3.4	2–3	2.4–3.4
4	3.6–4.6	3–3	0.91–1.16
12	6.9–7.9	5–6	0.58–0.66

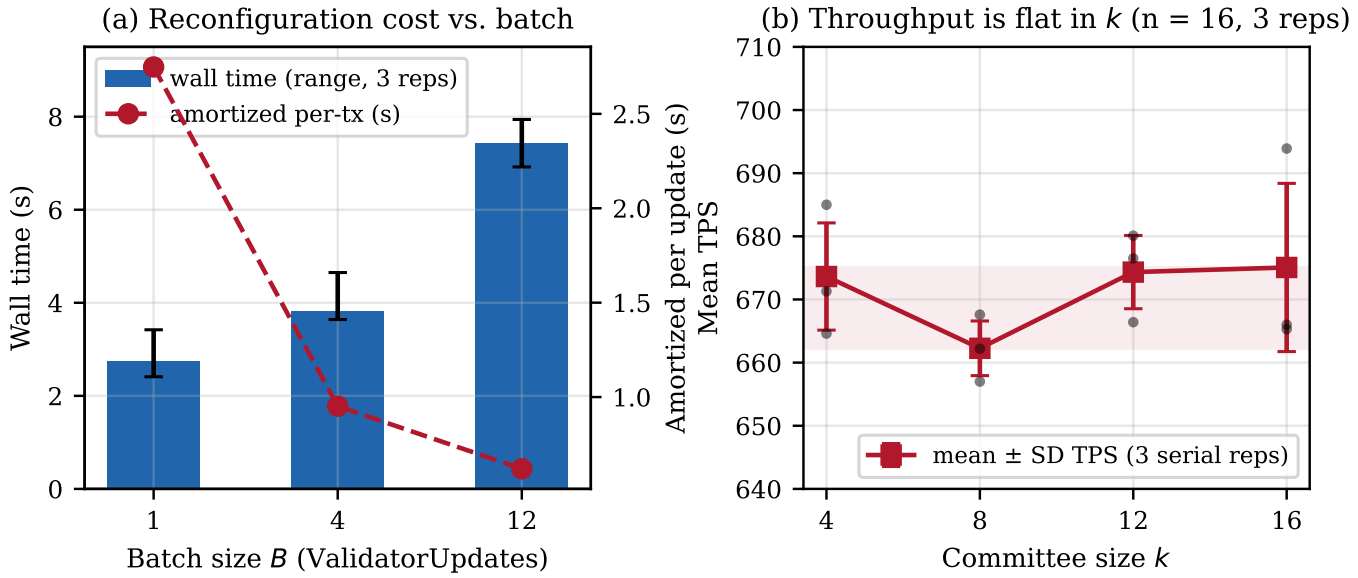


Figure 4. RQ2: (**a**) committee reconfiguration cost vs. batch size (three reps, range bars); (**b**) throughput vs. committee size k (three serial repetitions, mean ± SD with per-rep dots). The shaded horizontal band in (**b**) spans the min–max range of the four committee-size means.

Results and analysis. Reconfiguration cost is sublinear in batch size (amortized 3.4 s → 0.58 s per update as B grows 1→12, since updates piggyback on the same blocks), and a full 16→4 committee turnover (12 removals) completes in under 8 s, well within one harness epoch (13.7–16.6 s). BayesElect's actuation overhead is negligible at the control timescale, validating the epoch granularity design of Section 4. Two implementation guards proved necessary, and are now part of the design. First, a quorum guard caps removals at $n_{\text{active}}-4$, keeping the elected set at or above the BFT minimum $k_{\min} = 4$; a pre-guard version of this experiment silently drained the validator set towards empty and stalled the chain. Second, an intent ledger keyed by (public key, power) with expiry; keying by public key alone lets a stale shrink intent block a later restore of the same validator, which we observed to silently shrink the validator set before the fix. (b) Throughput is flat in k at $n = 16$: the four committee sizes measure 662.3–675.1 TPS (Table 7; three-repetition means; spread of means 1.9%), and the per-k ranges overlap almost completely. Pairing repetitions by index, $k = 16$ vs. $k = 8$, differs by +12.8 ± 17.2 TPS ($t = 1.29$, not significant). At this scale, CometBFT's capacity is block-production bound, not voting-quorum bound, so the small elected committee of RQ1 is obtained for free and the security margin of Theorem 1 costs no throughput.

Table 7. RQ2 committee capacity scan.

k	**4**	**8**	**12**	**16**
Mean TPS	673.6 ± 10.4	662.3 ± 5.3	674.3 ± 7.1	675.1 ± 16.3
Range (min …max)	664.6 …685.0	657.0 …667.6	666.4 …680.1	665.3 …693.9

4.3. RQ3: Silence Attack Resilience (Table 8 and Figure 5)

Configuration. Silence injected at $t = 300$ s, with the per-mode maximal liveness-preserving dose of Section 3.4.6 (1 of 4 committee members for AdaTrust; 3 of 16 voters for StaticTM), six paired runs of 600 s per mode interleaved on chain A; AdaTrust at $k = 4$ pre-attack.

In M6 follow-up (six runs, chain A, attack at $t = 300$ s), StaticTM pushed to its maximal liveness-preserving dose with 5 of 16 validators stopped, leaving exactly the quorum

minimum of 11 voters. This was measured in a separate campaign window, and only within-campaign figures are compared (Section 3.4.4).

Table 8. RQ3 real-chain results ($n = 6$ paired runs; paired t-tests, df = 5). Bold indicates the statistically better outcome per metric.

Metric	AdaTrust	StaticTM	Test
Retention (post/pre TPS)	**0.909 ± 0.037**	0.759 ± 0.080	$t = 3.17$, $p<0.05$, $d_z = 1.29$
Overall TPS	**630.8 ± 8.0**	570.6 ± 25.8	+10.5
Post-attack TPS (absolute)	**600.9 ± 16.1**	492.1 ± 55.8	+22.1
StaticTM @ max dose (5/16 silent)	—	retention 0.808 ± 0.047; post-attack 507.5 ± 12.4 TPS	separate campaign
Detection latency (s)	142.1	154.9	both detect (n.s.)
Healing: k 4→13 after detection	**6/6 runs**	0/6	—
Recovery to ≥80			

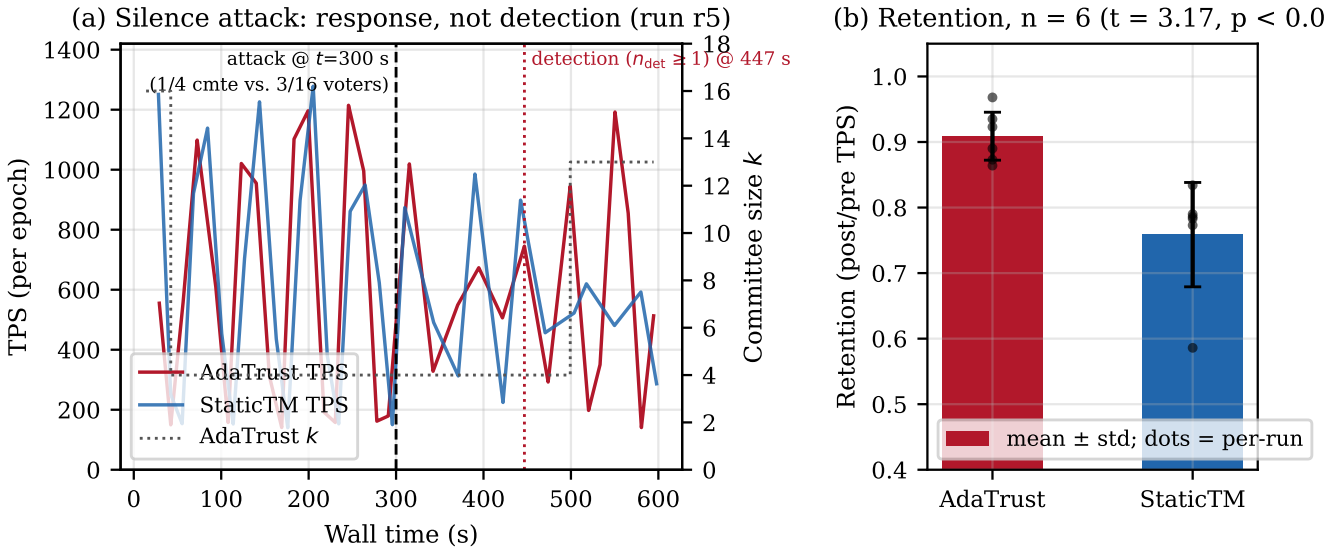


Figure 5. RQ3 silence attack timelines (attack at $t = 300$ s): (**a**) trust-score decay, detection, and committee healing ($k : 4 \rightarrow 13$) for AdaTrust vs. StaticTM (example run r5); (**b**) retention per mode ($n = 6$).

Results. Table 8 reports the per-run metrics. Both instrumentation planes detect the silence (142 s vs. 155 s, statistically indistinguishable); at $\lambda_d = 0.98$ per-block decay, a stopped validator's trust score crosses θ_{det} after ≈53 blocks (≈140 s at the measured block rate), matching the theory of Section 3.2 almost exactly. The divergence is in the response. AdaTrust's control plane uses $\hat{f}_{\text{eff}} = \max(\hat{f},\ n_{\text{det}}/16)$; the detection floors $\hat{f}_{\text{eff}}$ at 1/16, Equation (9) sizes $k \rightarrow 13$, and the committee re-elects within ≈51 s of detection in all 6 runs (49.4–52.4 s, ≈3 harness epochs), sidelining the dead committee member from the quorum's critical path. StaticTM keeps its three dead voters in the 16-voter set permanently and loses 24.1% of throughput; AdaTrust loses 9.1% (retention 0.909 vs. 0.759, $d_z = 1.29$; overall +10.5% TPS, $p < 0.01$). Dose check (M6): Because the 3/16 StaticTM dose is below the maximal liveness-preserving dose (5/16, Section 3.4.6), we re-ran StaticTM alone at five silent validators (six runs), resulting in retention 0.808 ± 0.047 and absolute post-attack service 507.5 ± 12.4 TPS versus AdaTrust's 600.9 ± 16.1 TPS. We report two aspects of this follow-up. (i) The headline comparison does not hinge on the lenient dose: at the maximal dose, the static chain still loses 19.2% of throughput where AdaTrust loses 9.1%, and serves 15.6% fewer transactions after the attack begins. (ii) Our earlier expectation that the higher dose would widen the retention gap was not confirmed: 5/16 retention (0.808) is if anything higher than 3/16 retention (0.759); the two campaigns ran in different windows, and with 11 voters exactly at quorum, the chain keeps saturating block production under the

offered load, so the dose response below the halt threshold is flat within drift. The robust discriminator is the response (healing), not the dose. Two further items deserve notice here: (i) Recovery as opposed to retention: at ≥80% of pre-attack movement, the strict average criterion is met within the 600 s window in only 3/6 runs for both modes (best 15.2 s); at this attack size and window, retention is the discriminator, not full recovery, and longer windows are left to future work. (ii) We encountered and fixed two traps. First, CometBFT v0.38's RPC returns ABSENT `last_commit` entries with an empty validator address; the signature array is positional, so an evidence pipeline keyed by address silently drops exactly the absent votes it exists to detect. Thus, we map entries positionally against the ordered validator set cached per `validators_hash`. Second, a $\hat{f}$ based on vote share is structurally self-defeating for sizing; a detected validator's weight collapses that so $\hat{f} \to 0$ precisely when detection succeeds, and a naive controller would keep the dead member forever. This justifies the n_{det} floor in $\hat{f}_{\text{eff}}$.

4.4. RQ4: Ablation (Table 9, Figure 6)

Configuration. Benign square wave ($P = 120$ s), six paired runs per variant interleaved on chain A; the w/o-switching-cost variant is a separately retrained policy that is evaluated offline (four seeds).

Table 9. RQ4 ablation ($n = 6$ paired runs; per-round paired t-tests vs. full, df = 5).

Variant	Evaluation	TPS (Mean ± SD)	Paired Δ vs. Full (95% CI)
Full AdaTrust	real chain	646.6 ± 18.8	—
w/o trust observations	real chain	635.8 ± 16.1	-10.7 ± 10.3, $t = -2.55$, $p \approx 0.05$, $d_z = -1.04$, $[-21.5, +0.1]$
w/o sizing (k frozen at 4)	real chain	648.3 ± 15.4	$+1.8 \pm 15.1$, $t = 0.29$, n.s., $[-14.0, +17.6]$
w/o hysteresis	real chain	640.9 ± 34.5	-5.6 ± 35.9, $t = -0.38$, n.s., $[-43.3, +32.1]$
w/o switching-cost ($\omega_4 = 0$, retrained)	offline (training env, 4 seeds)	degenerate static corner in **4/4 seeds** (protocol probability ≥93.7)	

Analysis. Table 9 reports the paired differences; with six paired runs per variant, the benign load picture is a nuanced one. Removing trust observations costs 10.7 TPS on average (paired $d_z = -1.04$), a borderline effect ($t = -2.55$, df = 5, $p \approx 0.05$, CI $[-21.5, +0.1]$); the trust features buy a small amount of benign throughput beyond their decisive role under attack (RQ3). Removing sizing changes nothing (+1.8 TPS, n.s.), consistent with RQ2(b)'s flatness at $n = 16$. Removing hysteresis leaves the mean statistically unchanged (−5.6 TPS, n.s.) but inflates the run-to-run variance 3.4-fold (SD 34.5 vs. 18.8; worst run 584.6). At this sample size, the oscillation cost of an ungated controller shows up as instability rather than as a mean shift, and is reported as such. The $\omega_4 = 0$ ablation (the only training-time variant) shows what the switching cost term does during learning. In all four seeds, PPO degenerates to a static single-protocol corner (≥93.7% protocol probability on one slot in both load phases), with terminal-window training environment throughput of 261.5 ± 186.6 versus 460.5 for the full retrain. Two issues deserve mention here. First, the corner identity varies across seeds (Tendermint ×3, HotStuff ×1) and severity is corner-dependent; since our testbed realizes both slots on CometBFT-family backends, this particular collapse would be benign on our pool, meaning that the term's protective value is pool-dependent. Second, the evidence is offline and in the training environment, so it represents evidence about training dynamics (the term prevents collapse and preserves the option to switch) rather than a measured real-chain loss. With the term present, the trained policy retains a calibrated willingness to switch, but exercises it zero times on this pool (which is the correct answer

here). The halt regime experiment of Section 4.5 shows that the same policy class switches exactly when switching is the only liveness path.

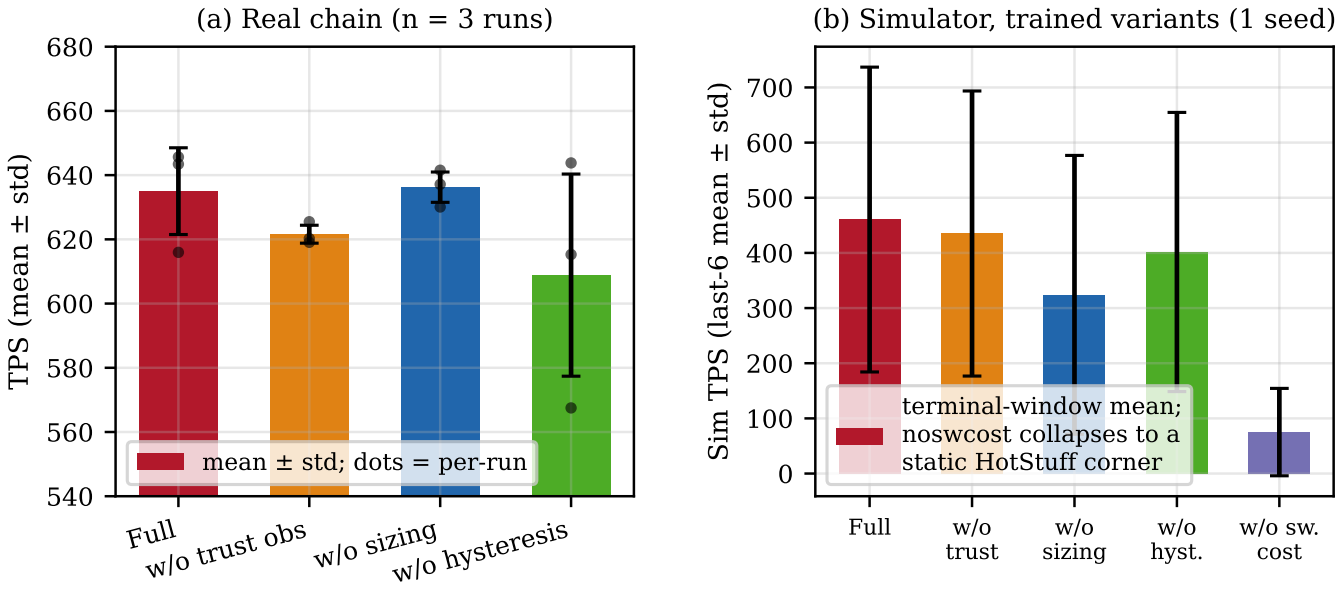


Figure 6. RQ4 ablation: (**a**) real-chain variants (n = 6 runs); (**b**) simulator-side terminal window throughput of the trained variants (one seed), including collapse to a static corner for the offline w/o-switch-cost variant.

4.5. RQ5: Halt-Regime Escape—Switching as the Only Liveness Path (Table 10)

Configuration. Halt injection at $t = 300$ s, 6 of 16 CometBFT validators stopped (`docker stop`), leaving ten voters below the $\lceil 2n/3 \rceil = 11$ quorum; chain A halts, and no committee re-election can restore it (the failed unit is the backend itself, not a removable member). Six paired runs of 600 s per mode, interleaved on chain A; both arms enter at $k = 16$ and serve ≈630–646 TPS before the halt (within-campaign parity). AdaTrust carries the halt regime controller, consisting of the same actor–critic architecture and observation as RQ1–RQ4, warm-started by behavioral cloning of a privileged teacher (hold the Tendermint lane while it serves; hold the QBFT lane while Tendermint is down) and fine-tuned by PPO under a permanent backend-outage curriculum. PPO fine-tuning did not improve held-out escape behavior, so the distilled checkpoint is the one deployed here. StaticTM is the control.

Table 10. RQ5 halt regime results ($n = 6$ paired runs).

Metric	AdaTrust	StaticTM	Test
Pre-halt TPS ($t < 300$ s)	646.0 ± 18.8	630.8 ± 14.7	parity
Liveness after halt (any commit)	**6/6**	0/6	—
Escape latency (injection → first QBFT block)	**95.1 ± 5.6 s**	—	—
Post-halt service (TPS)	**115.6 ± 24.4**	0.0 (cumulative frozen)	—
Overall TPS (full 600 s)	**347.5 ± 4.0**	315.1 ± 9.9	+32.4 ± 9.4, $t = 7.66$, $p < 10^{-3}$

Analysis. Table 10 reports the halt regime metrics. On the benign pool, the controller's calibrated answer is to never switch (RQ1, RQ4); RQ5 constructs the regime in which switching is the only way to stay alive, and the same policy class exercises the option. With 6 of 16 voters stopped, the Tendermint backend halts permanently ($10 < 11$); the trust loop detects the crash exactly as in RQ3, but committee healing cannot help because the backend itself, not a member, is the failed unit. In 6/6 runs, AdaTrust escapes to the QBFT backend after 95.1 ± 5.6 s—a few zero-service epochs on the halted chain to trip the controller's halt signal, followed by the measured handover drain—and restores 115.6 ± 24.4 TPS of

degraded but nonzero service for the remainder of the window. StaticTM serves zero once its cumulative commit counter freezes, and its overall throughput falls to 315.1 versus AdaTrust's 347.5 TPS (paired +32.4, $t = 7.66$, df = 5). Two notes are necessary. First, the escape lands on a slower lane, not a faster one; post-halt service is ≈18% of the pre-halt 630 TPS, and is ramp-limited by the fresh-JVM QBFT warm-up documented in Section 4.1. RQ5 supports the claim of liveness under backend failure, but does not support a throughput gain (a slow QBFT lane is strictly better than a dead Tendermint lane). Second, the mechanism is measured; the escape latency decomposes into zero-service detection on the halted chain, hysteresis dwell, and instrumented handover drain, all of which are recorded per epoch. Together with RQ4, the two experiments bracket the switching lever, which correctly refuses to switch when the pool offers no benefit and correctly switches when not doing so would mean zero service.

4.6. Controller Overhead and Reproducibility

The DRL controller's inference cost (≈3×10^5 FLOPs per forward pass) is one to two orders of magnitude below the per-round signature verification budget at $k = 4$ and constant in n (Section 3.3.3). Trust updates are $O(n)$ per block and measured reconfiguration completes in <8 s (Table 6), all negligible against the 13.7–16.6 s control epoch. Training is offline and one-time (≈480–580 s per seed on one H100 in the calibrated simulator).

Reproducibility statement. All raw epoch-level records are in `runs/real/rq1..5/` (per-run CSVs with per-epoch chain, committee, trust, detection, and backlog fields); experiment drivers and analysis scripts are in `code/prototype/` (`rq1_*_driver.sh`, `rq2_committee_scaling.py`, `rq3_silence_driver_v3.sh`, `rq4_ablation_driver.sh`, `analyze_rq3.py`); figures regenerate from the CSVs at 300 DPI in PNG and PDF [49]. The testbed recipe (CometBFT v0.38.17 16-container network, Besu 24.12.2 QBFT network, harness) is documented in the repository, including the positional signature mapping and quorum guard fixes of Sections 4.2 and 4.3, without which the measurements cannot be reproduced.

5. Discussion

5.1. Applicability Boundaries

AdaTrust targets permissioned/consortium settings with a registered validator set. Three boundaries deserve emphasis. First, the trust layer presupposes persistent identities: in permissionless networks where validators freely re-enter, Beta posteriors can be reset by re-registration, and BayesElect's guarantee degrades to uniform sampling unless identity-binding is added. Second, the value of small committees diminishes with thousands of validators, since random sortition already achieves low corruption probability. Third, as AdaptSwitch operates at epoch granularity, workloads oscillating within a single epoch cannot be tracked without multiplying the switching overhead.

5.2. Deployment Challenges

- **Cold start.** The policy and trust posteriors both require history. Deployment can be done in shadow mode while posteriors warm up (Lemma 1 gives the required evidence count). Two conditions must be budgeted separately because their detection times differ by orders of magnitude: a previously well-behaved validator that turns silent crosses θ_{det} after ≈53 blocks (≈140 s, Section 4.3), whereas a validator that is silent from genesis crosses after ≈1 block.
- **Cross-platform heterogeneity.** Validators on different hardware experience different verification costs; thus, deployments should estimate per-validator budgets to avoid the timeout action being miscalibrated for the slowest correct quorum members.

- **Controller governance.** A compromised controller cannot break safety (Theorems 1 and 2 hold for any policy), but can degrade performance arbitrarily; therefore, policy updates should follow the same governance process as protocol upgrades.

5.3. Limitations

1. **Scale.** Real-chain measurements are at $n = 16$ validators on one machine. The mechanisms transfer, but absolute TPS figures are hardware-specific. Multi-host testnets and equivocation injection remain future work.
2. **Sim-to-real transfer is per-deployment.** The policy is trained in a calibrated simulator; RQ4 shows that a retrained policy can chase a simulator niche that the real pool does not have. Each deployment must recalibrate and revalidate in shadow mode.
3. **Shared machine drift.** Dual-stream campaigns oversubscribe the 32 cores; we mitigated this via interleaved pairing, but cross-stream absolutes are incomparable and $n = 3$ campaigns have wider intervals.
4. **Attack and recovery coverage.** Live injection covers silence only; equivocation, larger coordinated doses, and longer windows remain future work. The strict ≥80% recovery criterion was met within 600 s in only half the runs for both modes. Our claim here is about retention and healing response, not full recovery.
5. **Adversary model coverage.** We did not cover long-range trust history attacks, adaptive reward poisoning beyond the bounded model (Assumption B4), or eclipse attacks. Theorem 5 scales linearly with ρ and offers no protection as $\rho \to 1$.

5.4. Future Work

1. **Scale-out testnets** at $n \in \{64, 256\}$ and on-chain double-sign injection, both enabled by the harness built here.
2. **Proposer quality trust features**: The cost of differing proposers for small committees under sustained load (Section 3.4.2) motivates extending the trust layer with proposer latency evidence and load-aware $k_{\min}$.
3. **Constrained MDP formulations** (e.g., Lagrangian PPO) to enforce security as hard limits rather than penalty shaping; robust Bayesian estimators against trust history attacks.

6. Conclusions

This paper addresses a structural limitation of deployed blockchain consensus, namely, that protocols are static while the environments they serve are not. We present AdaTrust, a dual-loop architecture in which Bayesian trust governs committee membership and a PPO controller governs protocol identity and configuration; the two loops are coupled through a handover mechanism with a formal safety invariant. The analysis establishes a configurable bound on committee corruption, proves that safety survives protocol switches, and characterizes the convergence and robustness of the learned policy under bounded adversarial influence. All performance claims were measured on live chains (a 16-validator CometBFT network and a 4-validator Besu QBFT network), with the simulator used only for training. The results confirm and sharpen our thesis: a trust-driven $k = 4$ committee matches full-set Tendermint at a quarter of the voting set, achieves 7.8× the matched-workload QBFT throughput, and heals the committee around a crashed member in all six attack runs, retaining 90.9% of throughput where the static chain keeps 75.9%. The switching lever pays only where structural niches differ; on our pool, the trained controller correctly refuses to switch, and our ablation shows that this refusal is earned by the switching cost term. When the Tendermint backend itself is removed, the same controller that refuses on the benign pool escapes to QBFT in 6/6 runs (95.1 s), restoring 115.6 TPS of degraded service against the static chain's zero. At the same time, real deployment feed back into the design

by surfacing effects that simulations do not produce. The broader lesson stands: harvesting the diversity of modern consensus designs requires controllers that learn on real evidence which levers pay.

Author Contributions: Conceptualization, S.T. and Z.J.; methodology, S.T.; software, S.T.; validation, S.T. and Z.W.; formal analysis, S.T.; investigation, S.T. and Z.W.; resources, Z.J.; data curation, S.T.; writing—original draft preparation, S.T.; writing—review and editing, S.T., Z.J., and Z.W.; visualization, S.T.; supervision, Z.J.; project administration, Z.J.; funding acquisition, Z.J. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Hebei Provincial Science and Technology Program Project (Grant No. 25360301D): Research and Application Demonstration of Key Technologies for Public Data Authorization and Operation Based on Trusted Data Space.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The source code and raw data that support the findings of this study are openly available in AdaTrust at https://github.com/tangsng/AdaTrust (accessed on 18 October 2026).

Acknowledgments: During the preparation of this manuscript, the authors did not use AI or AI-assisted tools for any aspect of the work.

Conflicts of Interest: The authors declare no conflicts of interest; the funders had no role in the design of the study, in the collection, analysis, or interpretation of data, in the writing of the manuscript, or in the decision to publish the results.

Abbreviations

The following abbreviations are used in this manuscript:

BFT	Byzantine Fault Tolerance
DRL	Deep Reinforcement Learning
PPO	Proximal Policy Optimization
TPS	Transactions Per Second
MDP	Markov Decision Process

Appendix A. Artifact Pointers and Supplementary Statistics

Appendix A.1. Block Capacity Evidence (Section 4.1)

The largest physically committed QBFT block in our campaign contains 1750 simple transfers at height 16,439 (≈36.75M gas at 21,000 gas/transfer), recorded in `runs/real/inj/inj_besu_1788823548.csv` ($t = 463.8$ s, `block_ntxs` column). By contrast, the saturated steady state under admission throttling averages 16 tx/block with `pending` ≤250. Both figures come from the same injector CSVs, which log per-second committed height, transactions per block, pool depth, and injected count.

Appendix A.2. Drain Time Distribution (Table 5)

The B3 reactive switcher's 15 handover drains average 10.8 s with standard deviation 9.4 s (coefficient of variation 87%): drains are short (2–4 s) when the switch lands near a phase boundary with an empty queue and long (up to 31 s) when a full QBFT backlog must be replayed onto CometBFT. The switcher uses dwell $d_{\min} = 5$ s and threshold 600 tx/s; drain is measured as handover start to first committed block on the target chain. The high variance is itself an argument for the hysteresis gate, as the cost of a switch is both large and unpredictable.

Appendix A.3. Statistical Unit Note (Table 5)

An earlier draft reported paired tests over 30 s phase-matched buckets ($n \approx 118$–121 per pair); because consecutive buckets on a live chain are autocorrelated, Table 5 reports run-level paired tests ($n = 6$ interleaved pairs per comparison), which are conservative by construction. The bucket-level analysis agrees in sign and magnitude on all rows, and is retained in the project repository (`rq1_summary.md`).

Appendix A.4. Software Versions and Reproducibility

Testbed: CometBFT v0.38.17 (16 Docker containers per chain; chains A/B on port ranges 267xx/277xx), Hyperledger Besu 24.12.2 (four JVM validators, QBFT), Python 3.10, PyTorch 2.10 (CUDA 12.8), web3.py injector with `tx-pool-max-future-by-sender` at its default 200 unless stated. Hardware: single GPU node (2× NVIDIA H100 PCIe 80 GB, 32 physical cores, 251 GiB RAM); both chains and the load generators share this machine, which is why all comparisons are constructed within-stream (Section 3.4.4). Code, harness scripts, raw CSVs, and analysis notebooks are openly available in the AdaTrust repository at https://github.com/tangsng/AdaTrust (accessed on 18 October 2026).

References

1. Islam, S.; Islam, M.J.; Hossain, M.; Noor, S.; Kwak, K.-S.; Islam, S.M.R. A survey on consensus algorithms in blockchain-based applications: Architecture, taxonomy, and operational issues. *IEEE Access* **2023**, *11*, 39066–39082. [CrossRef]
2. Castro, M.; Liskov, B. Practical Byzantine fault tolerance and proactive recovery. *ACM Trans. Comput. Syst.* **2002**, *20*, 398–461. [CrossRef]
3. Yin, M.; Malkhi, D.; Reiter, M.K.; Gueta, G.G.; Abraham, I. HotStuff: BFT consensus with linearity and responsiveness. In *Proceedings of the PODC'19: ACM Symposium on Principles of Distributed Computing, Toronto ON Canada, 29 July–2 August 2019*; Association for Computing Machinery: New York, NY, USA, 2019; pp. 347–356. [CrossRef]
4. Ge, L.; Wang, J.; Zhang, G. Survey of consensus algorithms for proof of stake in blockchain. *Secur. Commun. Netw.* **2022**, *2022*, 2812526. [CrossRef]
5. Danezis, G.; Kokoris-Kogias, L.; Sonnino, A.; Spiegelman, A. Narwhal and Tusk: A DAG-based mempool and efficient BFT consensus. In *Proceedings of the EuroSys'22: Seventeenth European Conference on Computer Systems, Rennes, France, 5–8 April 2022*; Association for Computing Machinery: New York, NY, USA, 2022; pp. 34–50. [CrossRef]
6. Spiegelman, A.; Giridharan, N.; Sonnino, A.; Kokoris-Kogias, L. Bullshark: DAG BFT protocols made practical. In *Proceedings of the CCS'22: 2022 ACM SIGSAC Conference on Computer and Communications Security, Los Angeles, CA, USA, 7–11 November 2022*; Association for Computing Machinery: New York, NY, USA, 2022; pp. 2705–2718. [CrossRef]
7. Gelashvili, R.; Kokoris-Kogias, L.; Sonnino, A.; Spiegelman, A.; Xiang, Z. Jolteon and Ditto: Network-adaptive efficient consensus with asynchronous fallback. In *Proceedings of the International Conference on Financial Cryptography and Data Security, Grenada, Grenada, 2 May 2022*; Springer: Berlin/Heidelberg, Germany, 2022; pp. 296–315. [CrossRef]
8. Gao, Y.; Lu, Y.; Lu, Z.; Tang, Q.; Xu, J.; Zhang, Z. Dumbo-NG: Fast asynchronous BFT consensus with throughput-oblivious latency. In *Proceedings of the CCS'22: 2022 ACM SIGSAC Conference on Computer and Communications Security, Los Angeles CA, USA, 7–11 November 2022*; Association for Computing Machinery: New York, NY, USA, 2022; pp. 1187–1201. [CrossRef]
9. Tan, H.; Gu, S.; Zhu, H.; Shen, J.; Bhuiyan, Z.A.; Wu, Q.M.J. An efficient ASCON-based group authentication and key agreement scheme with non-linkability and integrity assurance for IIoTs. *IEEE Trans. Dependable Secur. Comput.* **2026**, *23*, 8032–8047. [CrossRef]
10. Sourav; Ali, R.; Obaidat, M.S. Lattice-based certificateless signcryption scheme using blockchain for smart healthcare systems. *IEEE Trans. Emerg. Top. Comput.* **2026**, *early access*. [CrossRef]
11. Zhang, M.; Li, J.; Chen, Z.; Chen, H.; Deng, X. An efficient and robust committee structure for sharding blockchain. *IEEE Trans. Cloud Comput.* **2023**, *11*, 2562–2574. [CrossRef]
12. Hou, C.; Zhou, M.; Ji, Y.; Daian, P.; Tramèr, F.; Fanti, G.; Juels, A. SquirRL: Automating attack analysis on blockchain incentive mechanisms with deep reinforcement learning. In *Proceedings of the 28th Network and Distributed System Security (NDSS) Symposium 2021, Virtual, 21–25 February 2021*; Internet Society: Fredericksburg, VA, USA, 2021. [CrossRef]
13. Berger, C.; Reiser, H.P.; Sousa, J.; Bessani, A. AWARE: Adaptive wide-area replication for fast and resilient Byzantine consensus. *IEEE Trans. Dependable Secur. Comput.* **2022**, *19*, 1605–1620. [CrossRef]
14. Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; Klimov, O. Proximal policy optimization algorithms. *arXiv* **2017**, arXiv:1707.06347. [CrossRef]

15. Ressi, D.; Romanello, R.; Piazza, C.; Rossi, S. AI-enhanced blockchain technology: A review of advancements and opportunities. *J. Netw. Comput. Appl.* **2024**, *225*, 103858. [CrossRef]
16. Zhang, J.; Cai, K.; Wen, J. A survey of deep learning applications in cryptocurrency. *iScience* **2024**, *27*, 108509. [CrossRef]
17. Li, C.; Qiu, W.; Li, X.; Liu, C.; Zheng, Z. A dynamic adaptive framework for practical Byzantine fault tolerance consensus protocol in the Internet of Things. *IEEE Trans. Comput.* **2024**, *73*, 1669–1682. [CrossRef]
18. Diallo, N.; Xu, L.; Alsagheer, D.; Lu, Y.; Shi, L. Optimized consensus with DAGWise: A GNN-enhanced approach for scalable and fault-tolerant DAG-based BFT. In *Proceedings of the 2025 IEEE International Conference on Blockchain and Cryptocurrency (ICBC), Pisa, Italy, 2–6 June 2025*; IEEE: New York, NY, USA, 2025; pp. 1–5. [CrossRef]
19. Chen, H.; Fu, X.; Yuan, Q.; Zhuang, Z.; Kang, J.; Liu, Z.; Wang, J.; Niyato, D. Trust model-based consensus optimization for vehicle platooning networks: A novel deep reinforcement learning approach with GenAI. *IEEE Trans. Intell. Transp. Syst.* **2025**, *26*, 11356–11371. [CrossRef]
20. Xu, C.; Zhang, P.; Yu, H.; Li, Y. Dynamic blockchain-empowered trustworthy end-edge collaborative computing via rotating multi-agent DRL. *IEEE Trans. Wirel. Commun.* **2025**, *24*, 4864–4878. [CrossRef]
21. Gutierrez, R.; Villegas-Ch, W.; Govea, J. Adaptive consensus optimization in blockchain using reinforcement learning and validation. *Front. Artif. Intell.* **2025**, *8*, 1672273. [CrossRef]
22. Chen, J.; Shi, L.; Huang, Q.; Wang, T.; He, D. On designs of decentralized reputation management for permissioned blockchain networks. *Comput. Model. Eng. Sci.* **2024**, *139*, 1755–1773. [CrossRef]
23. Rawlins, C.C.; Jagannathan, S.; Nadendla, S.S.V.I. A reputation system for provably-robust decision making in IoT blockchain networks. *IEEE Internet Things J.* **2024**, *11*, 14088–14099. [CrossRef]
24. Huang, C.; Wang, Z.; Chen, H.; Hu, Q.; Zhang, Q.; Wang, W.; Guan, X. RepChain: A reputation-based secure, fast, and high incentive blockchain system via sharding. *IEEE Internet Things J.* **2021**, *8*, 4291–4304. [CrossRef]
25. Chen, X.; Xue, G.; Yu, R.; Wu, H.; Wang, D. A vehicular trust blockchain framework with scalable Byzantine consensus. *IEEE Trans. Mob. Comput.* **2024**, *23*, 4440–4452. [CrossRef]
26. Wang, X.; Guan, Y. A hierarchy Byzantine fault tolerance consensus protocol based on node reputation. *Sensors* **2022**, *22*, 5887. [CrossRef]
27. Qin, H.; Guan, Y. Joint reputation based grouping and hierarchical Byzantine fault tolerance consensus protocol. *IEEE Access* **2023**, *11*, 90335–90344. [CrossRef]
28. Qi, J.; Guan, Y. Practical Byzantine fault tolerance consensus based on comprehensive reputation. *Peer-to-Peer Netw. Appl.* **2023**, *16*, 420–430. [CrossRef]
29. Wu, J.; Xia, C.; Chen, Q.; Yan, H. Reputation-enhanced leader election for blockchain consensus. In *Proceedings of the 2025 IEEE International Conference on Blockchain Technology and Information Security (ICBCTIS), Haikou, China, 5–7 December 2025*; IEEE: New York, NY, USA, 2025; pp. 1–6. [CrossRef]
30. Aublin, P.-L.; Guerraoui, R.; Knežević, N.; Quéma, V.I.; Vukolić, M. The next 700 BFT protocols. *ACM Trans. Comput. Syst.* **2015**, *32*, 12. [CrossRef]
31. Bahsoun, J.-P.; Guerraoui, R.; Shoker, A. Making BFT protocols really adaptive. In *Proceedings of the 2015 IEEE International Parallel and Distributed Processing Symposium, Hyderabad, India, 25–29 May 2015*; IEEE: New York, NY, USA, 2015; pp. 904–913. [CrossRef]
32. Wu, C.; Qin, H.; Amiri, M.J.; Loo, B.T.; Malkhi, D.; Marcus, R. BFTBrain: Adaptive BFT consensus with reinforcement learning. In *Proceedings of the NSDI'25: 22nd USENIX Symposium on Networked Systems Design and Implementation, Philadelphia, PA, USA, 28–30 April 2025*; USENIX Association: Berkeley, CA, USA, 2025; pp. 1563–1583. [CrossRef]
33. Qiu, L.; Xiao, J.; Shao, Z. Mechanized safety and liveness proofs for the Mysticeti consensus protocol under the LiDO model. In *Proceedings of the 47th IEEE Symposium on Security and Privacy, San Francisco, CA, USA, 18–21 May 2026*; IEEE: New York, NY, USA, 2026; pp. 1842–1861. [CrossRef]
34. Nakamoto, S. Bitcoin: A Peer-to-Peer Electronic Cash System. 2008. Available online: https://bitcoin.org/bitcoin.pdf (accessed on 18 October 2026).
35. Dwork, C.; Lynch, N.; Stockmeyer, L. Consensus in the presence of partial synchrony. *J. ACM* **1988**, *35*, 288–323. [CrossRef]
36. Hwerbi, K.; Amdouni, I.; Adjih, C.; Saidane, L.A.; Laouiti, A. Delay analysis of BFT consensus: Case study of Narwhal and Bullshark protocols. *Comput. Commun.* **2025**, *242*, 108278. [CrossRef]
37. Yin, H.; Zhou, C.; Wen, W.; Liu, Y. Fountain: DAG-based separate BFT consensus made Hashgraph practical. *IEEE Trans. Netw. Serv. Manag.* **2025**, *22*, 4126–4136. [CrossRef]
38. Buchman, E.; Kwon, J.; Milosevic, Z. The latest gossip on BFT consensus. *arXiv* **2018**, arXiv:1807.04938. arXiv.1807.04938. [CrossRef]
39. Ongaro, D.; Ousterhout, J. In search of an understandable consensus algorithm. In *Proceedings of the USENIX ATC'14: Proceedings of the 2014 USENIX conference on USENIX Annual Technical Conference, Philadelphia, PA, USA, 19–20 June 2014*; USENIX Association: Berkeley, CA, USA, 2014; pp. 305–319.

40. Gilad, Y.; Hemo, R.; Micali, S.; Vlachos, G.; Zeldovich, N. Algorand: Scaling Byzantine agreements for cryptocurrencies. In *Proceedings of the SOSP'17: ACM SIGOPS 26th Symposium on Operating Systems Principles, Shanghai, China, 28 October 2017*; Association for Computing Machinery: New York, NY, USA, 2017; pp. 51–68. [CrossRef]
41. Issa, W.; Moustafa, N.; Turnbull, B.; Sohrabi, N.; Tari, Z. Blockchain-based federated learning for securing Internet of Things: A comprehensive survey. *ACM Comput. Surv.* **2023**, *55*, 1–43. [CrossRef]
42. Li, J.; Xie, T.; Liu, Y. Game theory based incentive mechanism for a bi-tiered public blockchain system. *Comput. Netw.* **2026**, *274*, 111827. [CrossRef]
43. Cam, N.T.; Kiet, V.I.T. FlwrBC: Incentive mechanism design for federated learning by using blockchain. *IEEE Access* **2023**, *11*, 107855–107866. [CrossRef]
44. Sokhankhosh, A.; Rouhani, S. In *Proceedings of the 2024 IEEE International Conference on Blockchain (Blockchain), Copenhagen, Denmark, 19–22 August 2024*; IEEE: New York, NY, USA, 2024; pp. 370–377. [CrossRef]
45. Alam, S.; Zardari, S.; Shamsi, J.A. Blockchain-based trust and reputation management in SIoT. *Electronics* **2022**, *11*, 3871. [CrossRef]
46. Putra, G.D.; Kang, C.; Kanhere, S.S.; Hong, J.W.-K. DeTRM: Decentralised trust and reputation management for blockchain-based supply chains. In *Proceedings of the 2022 IEEE International Conference on Blockchain and Cryptocurrency (ICBC), Shanghai, China, 2–5 May 2022*; IEEE: New York, NY, USA, 2022; pp. 1–5. [CrossRef]
47. Drineas, P.; Nema, R.; Ostrovsky, R.; Zikas, V.I. Game of Trust: How trustworthy does your blockchain think you are? *arXiv* **2025**, arXiv:2505.14551. [CrossRef]
48. Pawar, P.; Kasula, V.K.; Bhuvanesh, A.; Kumar, D.; Yadulla, A.R.; Keerthanadevi, R. Exploring blockchain-enabled secure storage and trusted data sharing mechanisms in IoT systems. In Proceedings of the 2025 IEEE International Conference on Interdisciplinary Approaches in Technology and Management for Social Innovation (IATMSI), Gwalior, India, 6–8 March 2025. [CrossRef]
49. Touloupou, M.; Themistocleous, M.; Iosif, E.; Christodoulou, K. A systematic literature review toward a blockchain benchmarking framework. *IEEE Access* **2022**, *10*, 70630–70644. [CrossRef]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.