Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

Consensus among Learning Agents: A Multi-AgentReinforcement Learning Framework withGame-Theoretic Incentives

Chen Song
National University of Defense Technology
Jing Li
National University of Defense Technology
Tao Xie
hamishxie@vip.sina.com

National University of Defense Technology

Research Article

Keywords: Multi-agent reinforcement learning, Blockchain consensus, Game-theoretic mechanism design, Proximal Policy Optimization, Incentive compatibility, Autonomous agents

Posted Date: July 8th, 2026

DOI: https://doi.org/10.21203/rs.3.rs-10013916/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: No competing interests reported.

Consensus among Learning Agents: A Multi-Agent Reinforcement Learning Framework with Game-Theoretic Incentives

Chen Song[1], Jing Li[1], Tao Xie[1*]

[1]College of Computer Science and Technology, National University of Defense Technology, Changsha, 410073, China.

*Corresponding author(s). E-mail(s): hamishxie@vip.sina.com;
Contributing authors: songchen_25@nudt.edu.cn; lijing18@nudt.edu.cn;

Abstract

Consensus protocols for blockchain systems are typically designed around a fixed participant model: rational miners, rational stakers, or protocol-scheduled validators whose strategy space is bounded by rules set offline. As autonomous agents increasingly act on behalf of humans in decentralized settings, the participants themselves become *learning* entities whose policies drift during operation, falling outside the equilibrium assumptions of existing protocols. This paper studies consensus in such a setting. We cast the problem as a Markov game played by node agents, each trained with independent Proximal Policy Optimization, coupled to a game-theoretic reward mechanism whose constraints align individual incentives with role-appropriate behaviour, and implement the framework as a simulation environment of up to 100 nodes with deterministic per-round finality and longest-chain fork resolution. Across five research questions, we report that policies converge quickly, reward distribution is equitable across nodes after convergence, and adversarial tolerance aligns with the protocol's $\lfloor \boldsymbol{n/3} \rfloor$ congestion threshold; a decomposition into finality latency, liveness, and fork-resolution time exposes degradation modes that conventional success-rate-only views conceal. We also report three negative findings: a language-model state encoder yields no measurable gain over a numeric baseline while inflating training cost; the apparent benefit of a token-economy module is fully accounted for by exogenous transaction-pool injection; and an online adaptive parameter controller, evaluated against calibrated fixed baselines, does not improve steady-state performance or robustness, its sole consistent benefit being equivalent to

a reward-annealing schedule. We replace the framework's original probability-one convergence claim with an ε-approximate Nash existence result and identify centralized-critic variants as a direction for tightening the non-stationarity gap.

Keywords: Multi-agent reinforcement learning, Blockchain consensus, Game-theoretic mechanism design, Proximal Policy Optimization, Incentive compatibility, Autonomous agents

1 Introduction

Complex collective behaviour can emerge from large populations of simple, locally interacting units: bee colonies and bird flocks solve global problems through decentralized coordination, and modern large language models exhibit analogous emergent capabilities as their scale grows [1]. As autonomous agents built on such models begin to act, transact, and coordinate on behalf of humans, we anticipate decentralized agent societies in which self-interested, *learning* participants must reach agreement without a central authority [2]. Reaching agreement among such participants is fundamentally a consensus problem—but one whose participants and operating conditions are non-stationary in a way that classical consensus was never designed to handle.

Established consensus mechanisms are built on *static* rules. Their core parameters—block intervals, difficulty targets, committee and epoch sizes—are fixed offline or adjusted only on human timescales, under the implicit assumption of a relatively stable environment [3]. Bitcoin, for instance, fixes a target block interval of roughly ten minutes and retargets mining difficulty only once every 2016 blocks (about two weeks); sharded and committee-based protocols similarly preset committee and epoch sizes that change slowly, if at all [4]. This assumption is increasingly untenable. Eyal and Sirer [5] showed that once participants are modelled as rational rather than altruistic, honest participation ceases to be a dominant strategy under Bitcoin's PoW, exposing the brittleness of static rules against strategic behaviour. Later designs improve throughput and latency—decoupling leader election from transaction packaging in Bitcoin-NG [6], or partitioning the network in sharded protocols such as Elastico [7]—yet their governing parameters remain preset and slow to adjust. When the participants are learning agents whose policies drift within the lifetime of the system, a controller that cannot adapt at the same timescale will repeatedly fall out of equilibrium.

Our goal is therefore not to outperform permissioned Byzantine Fault Tolerant (BFT) protocols on raw throughput in their own static, low-node-count regime; such comparisons conflate fundamentally different design objectives. We instead target what we call the *adaptivity gap*: sustaining incentive compatibility, liveness, and fair reward allocation when both agent strategies and environmental conditions change continuously. To this end we formalize the consensus process as a Markov game played by node agents, in which the system's incentive parameters are themselves adjusted online from performance feedback, so that desirable collective behaviour is actively maintained rather than statically assumed.

Concretely, each node is an autonomous agent that learns a block proposal/validation policy via independent Proximal Policy Optimization (PPO), coupled with a game-theoretic reward mechanism designed so that honest participation is an approximate equilibrium strategy. An adaptive parameter controller monitors throughput, fork rate, and consensus success rate, and adjusts reward/penalty weights, inactivity thresholds, and decay factors at runtime. Two auxiliary components—a lightweight token-economy model and a language-model-based state enhancement—are included to make the simulated environment and agent observations more realistic; they support, but are not essential to, the core consensus mechanism.

The main contributions of this paper are summarized as follows:

- We cast decentralized consensus among self-interested learning agents as a Markov game and design an incentive-compatible reward mechanism whose constraints are intended to make honest behaviour an equilibrium strategy under dynamic conditions, together with an analysis of equilibrium existence under the stated assumptions.
- We propose an adaptive parameter-control mechanism that tunes consensus incentives online from real-time performance feedback, replacing the offline-tuned constants of static protocols and allowing the system to track non-stationary agent behaviour.
- We integrate these elements into a single self-regulating framework and evaluate it on networks ranging from 5 to 100 nodes, characterizing throughput convergence, reward-distribution fairness, and graceful degradation under up to 50% adversarial participants.

Figure 1 shows the overall architecture of the adaptive multi-agent consensus framework.

2 Related Work

2.1 Blockchain Consensus Mechanisms

Blockchain consensus mechanisms ensure data consistency and system security among distributed nodes; innovations in consensus design directly determine a system's scalability, throughput, and confirmation latency. Current mainstream approaches have evolved along two primary directions.

The first follows the Nakamoto consensus paradigm, with Bitcoin's Proof of Work [8] as its foundation, and seeks performance through architectural innovation. Bitcoin-NG [6] and Prism [9] decouple leader election from transaction packaging; Ethereum's GHOST protocol [10] and Conflux [11] adopt Directed Acyclic Graph (DAG) structures to increase concurrency; ORIC [12] employs graph-chain ledgers for adaptive high throughput. Chainweb [13] interlinks multiple parallel chains for security, while OHIE [14] runs multiple independent chains with cross-chain references. Monoxide [15] and ORIC-Shard [16] introduce sharding for near-linear throughput scalability.

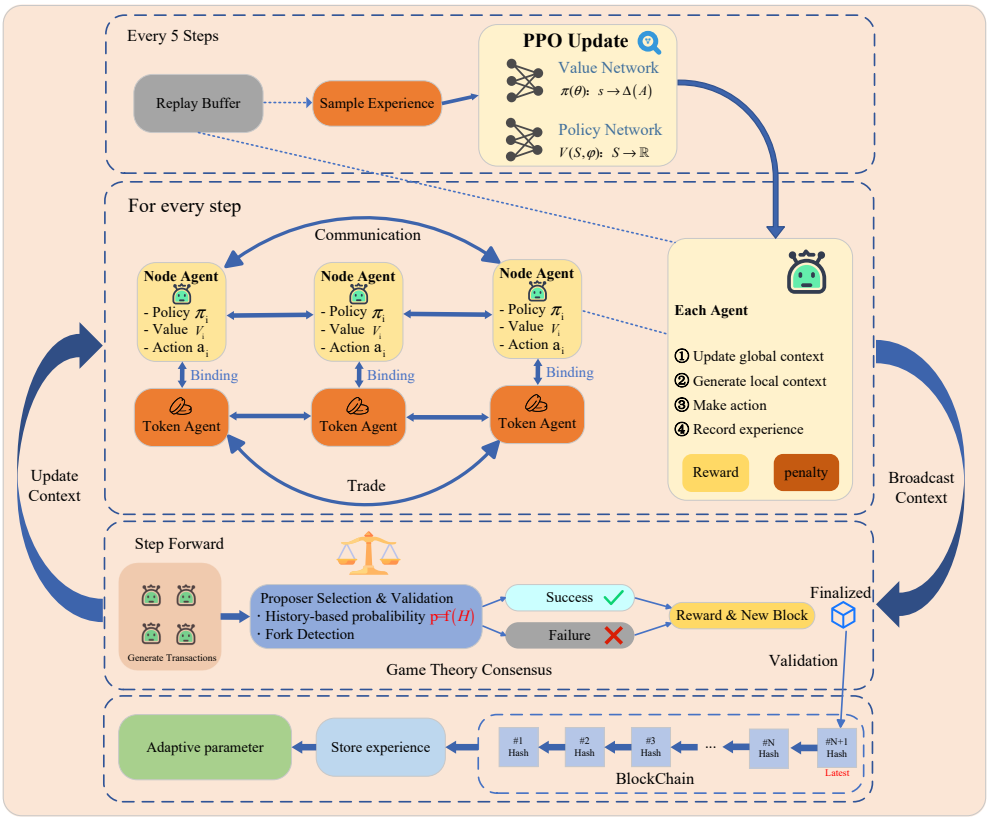


Fig. 1 Illustration of the Adaptive Multi-Agent Consensus Training Framework. The process is organized into six sequential stages: State Construction, Decentralized Decision Making, Game-Theoretic Consensus, Environment Update, Adaptive Parameter Control, and Learning. Node agents interact with the blockchain environment, update policies via PPO, and adaptively adjust system parameters.

The second direction is Byzantine Fault Tolerant (BFT) committee-based consensus, exemplified by Tendermint [17], adopted in Cosmos, whose primary advantage is fast finality. Such designs are often combined with sharding, as seen in Elastico [7], OmniLedger [18], and RapidChain [19]. These systems assign nodes to committees via Verifiable Random Functions [20], enabling parallel consensus while preserving BFT safety. Nakamoto-style mechanisms pursue security and decentralization in open environments through computational competition, whereas BFT mechanisms favor efficient finality through voting in more controlled settings.

Participant assumptions across the design space.

A property cutting across both directions, and one that motivates the present work, is the *participant model* these protocols assume. Nakamoto-style protocols assume rational miners whose strategy space is the set of mining/withholding decisions over fixed protocol rules; BFT protocols assume participants whose roles (proposer, validator)

are scheduled by the protocol; PoS protocols assume rational stakers within fixed staking rules. In each case the participant's strategy space is bounded by rules that do not change at the timescale of operation. Our setting is different: we study consensus among participants whose strategies are themselves *learned* and therefore drift during the system's lifetime, which falls outside the equilibrium assumptions of the protocols above.

2.2 Multi-Agent Reinforcement Learning

Tan [21] introduced *independent* Q-learning, in which each agent treats peers as part of the environment — a design simple, decentralized, and the starting point for many subsequent methods. A milestone for cooperative settings is MADDPG [22], which adopts the centralized training with decentralized execution (CTDE) paradigm: agents access global information during training but rely on local observations at execution time. Zhang et al. [23] survey MARL methodologies spanning game-theoretic foundations, value-function decomposition, and policy-gradient approaches.

For policy-gradient training specifically, Proximal Policy Optimization (PPO) [24] has become a standard single-agent algorithm and underpins our node-level learner. The natural multi-agent extension, MAPPO [25], pairs PPO with a CTDE centralized critic and has shown strong empirical performance on cooperative benchmarks. A long-standing concern with applying independent learners in multi-agent settings is non-stationarity: from any one agent's perspective, the environment changes as peers update their policies, weakening the convergence guarantees that motivate single-agent algorithms [26]. We adopt independent PPO in this work for its simplicity and decentralization properties, and discuss this trade-off explicitly in Section 3.3 and as a future direction.

MARL has been applied to a range of distributed-systems problems. Mao et al. [27] applied reinforcement learning to network routing, showing that learned policies outperform heuristics across diverse network conditions; the same group subsequently extended the approach to data-center task scheduling [28], learning resource-allocation strategies that surpass manually designed schedulers. Naderializadeh et al. [29] applied MARL to channel allocation and power control in wireless networks, demonstrating that distributed nodes can learn to coordinate and share resources through decentralized interactions. Closer to our methodological concerns, Li et al. [30] proposed a decentralized adaptive method with an explicit consensus step for non-convex non-concave min–max optimization over multi-agent networks, synchronizing adaptive learning rates across agents to mitigate divergence from inconsistent local updates. Although their target is optimization rather than blockchain consensus, the structural pattern — decentralized local updates coordinated by a consensus step — is the methodological neighbour of the present work.

2.3 Game-Theoretic Mechanism Design

Eyal and Sirer [5] provided a seminal analysis of strategic behaviour in Bitcoin's PoW consensus, showing rigorously that under a rational-miner model the "selfish

mining" strategy can yield higher rewards than honest behaviour. This finding challenged the assumption in Nakamoto consensus that honest participation is a dominant strategy, and highlighted the necessity of modelling consensus nodes as rational, self-interested game participants. In systems composed of such agents, mechanism design provides principled guidance on how to shape behaviour through carefully crafted rules — particularly reward functions — so that individual utility maximization aligns with system-level objectives, the property of *incentive compatibility* [31]. This line of reasoning directly informs the reward design in Section 3.2.

In the PoS setting, Saleh [32] analyses consensus from game-theoretic and financial-economic perspectives, examining the "nothing-at-stake" problem and its mitigations. The Ouroboros protocol [33] provides the first PoS protocol with a rigorous cryptographic security proof, integrating game-theoretic reasoning with cryptographic techniques under rational adversary models. Beyond cryptocurrencies, Zhang et al. [34] apply game-theoretic mechanism design to federated learning data trading, modelling participant incentives to ensure data quality and fair reward distribution.

Relation to this work.

Protocols such as Ouroboros enforce equilibrium behaviour through static, cryptographically guaranteed rules over a fixed participant model. We share the broader objective of aligning individual incentives with collective outcomes, and inherit the incentive-compatibility design principle of the works above. We differ in the participant model: our nodes are *learning* agents whose policies evolve during operation, which is the setting our framework targets. Whether dynamic adjustment of the incentive parameters themselves adds value beyond a well-calibrated static design is an empirical question, and we examine it directly in Section 4.5 rather than claiming it as an a priori advantage.

3 Methodology

We describe the framework in four parts: an overview of the training loop and its formal scaffolding (Section 3.1), the game-theoretic consensus mechanism that constitutes the core contribution (Section 3.2), the learner and an analysis of equilibrium existence (Section 3.3), and the systems-level mechanisms for adaptive control and fork resolution (Section 3.4). Two auxiliary modules — a token-economy load generator and an optional language-model state encoder — are summarized briefly in Section 3.5; the experimental record in Section 4 examines whether they earn their place.

3.1 Framework Overview

We model the multi-agent blockchain consensus system as a ten-tuple

$$S = \langle N, E, A, G, P, T, \Gamma, M, R, O \rangle, \tag{1}$$

where $N = \{n_1, \ldots, n_k\}$ is the set of nodes, each an autonomous decision-making agent; E is the blockchain environment that simulates transaction generation, block

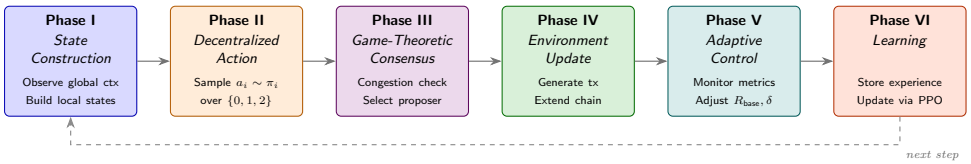


Fig. 2 The training loop is organized into six phases: state construction, decentralized action selection, game-theoretic consensus, environment update, adaptive parameter control, and policy/value learning. Node agents interact with the blockchain environment and update their policies via PPO; the adaptive controller adjusts incentive parameters online.

propagation, and network latency; A is the agent system encompassing each node's policy and learning algorithm; G is the game-theoretic mechanism that resolves proposer competition and assigns rewards; P is the adaptive controller that adjusts incentive parameters from performance feedback; T is the discrete time index; Γ is the consensus protocol including block validation and chain-selection rules; M is the inter-node message-passing mechanism; R is the per-node reward function; and O is the observation space. The network operates under partial synchrony with probabilistic message loss; each node is treated as a rational agent that learns to maximize its own discounted return.

The training loop (Algorithm 1; Figure 2) is organized into six phases per step.

Phase I — State Construction. The system maintains a global context ctx_{global} containing blockchain length, throughput, consensus latency, success rate, and a network-health indicator. Each node i constructs a local state $\mathbf{s}_{i,\text{local}} = [n_{tx}, \Delta t, h]^\top \in \mathbb{R}^3$ (transaction-pool size, time since last block, local health). Globally observable features $\mathbf{s}_{global} = [L, \theta_{tput}, f_{fork}, \rho_{success}, \tau_{latency}, h_{network}]^\top$ are exposed to the adaptive controller but not directly to the policy. An optional language-model encoder may concatenate a semantic embedding to $\mathbf{s}_{i,\text{local}}$ (Section 3.5); experiments use the numeric state by default.

Phase II — Decentralized Action Selection. Each node samples an action $a_i^t \sim \pi_i(\cdot \mid s_i^t; \theta_i)$ over the discrete set $A_i = \{0, 1, 2\} = \{\textsc{Propose}, \textsc{Validate}, \textsc{Wait}\}$. Nodes designated as forced proposers (Section 3.2) must propose to preserve liveness; all others act according to the learned policy.

Phase III — Game-Theoretic Consensus. The joint action profile passes through the consensus mechanism G, which classifies proposers as voluntary or forced, applies a congestion check, selects a successful proposer when the round succeeds, and assigns role-dependent rewards or penalties. Section 3.2 develops this in full.

Phase IV — Environment Update. Transactions are generated as $\text{num_tx} = \max(1, \text{Poisson}(\lambda \Delta t))$ and propagated probabilistically; the ledger is extended with the newly finalized block (at most one per round); system metrics are recomputed. Fork detection and resolution (Section 3.4) maintain eventual consistency across replicas.

Phase V — Adaptive Parameter Control. The controller compares moving averages of throughput, fork rate, and success rate against target values and, when the gap exceeds a stability threshold, adjusts the base reward, base penalty, inactivity

Algorithm 1 Multi-Agent Consensus Training Framework

Input: Number of nodes n, episodes E, steps per episode T
Output: Trained node agents $\{\mathcal{N}_1, \ldots, \mathcal{N}_n\}$
Initialize blockchain environment $\mathcal{E}(n)$
Initialize node agents $\{\mathcal{N}_i\}_{i=1}^{n}$ with policy π_{θ_i} and value V_{ϕ_i}
for episode $e = 1$ to E **do**
    $s_g \leftarrow \mathcal{E}.\textsc{Reset}()$
    for step $t = 1$ to T **do**
        $actions \leftarrow [\,]$
        for each node $\mathcal{N}_i$ **do**
            $s_i \leftarrow \mathcal{N}_i.\textsc{GetLocalState}(s_g)$
            $a_i \leftarrow \mathcal{N}_i.\textsc{ChooseAction}(s_i)$
            $actions.\textsc{append}(a_i)$
        end for
        $(s'_g, \mathcal{R}, done) \leftarrow \mathcal{E}.\textsc{Step}(actions)$
        for each node $\mathcal{N}_i$ **do**
            $s'_i \leftarrow \mathcal{N}_i.\textsc{GetLocalState}(s'_g)$
            $\mathcal{N}_i.\textsc{StoreExperience}(s_i, a_i, r_i, s'_i, done)$
        end for
        if $t \bmod K = 0$ **then**
            for each node $\mathcal{N}_i$ **do**
                $\mathcal{N}_i.\textsc{Learn}()$
            end for
        end if
        $s_g \leftarrow s'_g$
        if $done$ **then**
            break
        end if
    end for
end for
return $\{\mathcal{N}_i\}_{i=1}^{n}$

threshold, and decay factor. Details and an empirical evaluation are in Sections 3.4 and 4.5.

Phase VI — Learning. Every K steps, each node updates its policy and value networks via independent PPO from its replay buffer (Section 3.3; Algorithm 4).

3.2 Game-Theoretic Consensus Mechanism

At every step, nodes occupy one of four roles: *voluntary proposer*, *forced proposer*, *validator*, or *waiting node*. A node that has not voluntarily proposed for an extended period is designated as a forced proposer to guarantee liveness. The mechanism (Algorithm 3) is built around a parameterized reward structure, a congestion check, and a history-weighted proposer-selection rule.

Reward design and incentive compatibility.

Table 1 states the per-role rewards. Three design principles structure the parameters:

1. *Proposer-incentive hierarchy:* the selected proposer receives the highest reward ($\delta(t) \cdot R_{base}$), followed by unselected voluntary ($\alpha_{vol} \cdot R_{base}$) and forced proposers ($\alpha_{forced} \cdot R_{base}$), with $\delta(t) > \alpha_{vol} > \alpha_{forced}$.
2. *Spam deterrence:* when consensus fails because too many nodes propose simultaneously, voluntary proposers pay a penalty $-\lambda_{vol} \cdot P_{base}$ with $\lambda_{vol} > \lambda_{forced} > 1$; forced proposers are penalized more mildly to preserve liveness.

Table 1 Per-role reward structure. R_{base} and P_{base} are dynamically adjusted by the controller of Section 3.4.

Role	Success	Failure
Selected proposer	$R_{base} \cdot \delta(t)$	—
Voluntary proposer	$R_{base} \cdot \alpha_{vol}$	$-\lambda_{vol} \cdot P_{base}$
Forced proposer	$R_{base} \cdot \alpha_{forced}$	$-\lambda_{forced} \cdot P_{base}$
Validator	$R_{base} \cdot \beta$	$R_{base} \cdot \eta_{verify}$
Idle node	$R_{base} \cdot \gamma$	$R_{base} \cdot \eta_{wait}$

3. *Validation over idleness:* validators receive $\beta \cdot R_{base}$ and idle nodes $\gamma \cdot R_{base}$ with $\beta > \gamma$, making active validation strictly preferred to passive waiting.

These constraints align the reward function with system-level objectives so that role-appropriate behaviour is a best response under the corresponding state.

Congestion check and proposer selection.

Let $I_{prop} = \{i : a_i = 0\}$ be the set of proposing nodes and τ the congestion threshold (default $\lfloor n/3 \rfloor$, with a floor of 2). The consensus protocol Γ is

$$\Gamma(a_1, \ldots, a_n) = \begin{cases} \Psi(I_{prop}, t, H), & |I_{prop}| \leq \tau, \\ \perp, & \text{otherwise}, \end{cases} \tag{2}$$

where H records per-node histories (cumulative reward, inactivity count, forced-proposer flag) and Ψ is the proposer-selection function. When the round fails ($\perp$), the spam-deterrence penalties apply; when it succeeds, Ψ samples a proposer according to weighted probabilities (Algorithm 2):

$$p_i^{select} \propto p_i^{base} \cdot \alpha_i^{idle} \cdot \beta_i^{force}, \tag{3}$$

with $p_i^{base} = \max(0.1, R_i^{hist}/100)$, $\alpha_i^{idle} = \min(2.0, n_i^{idle}/5)$, and $\beta_i^{force} = 2.0$ if node i is a forced proposer (else 1.0). The minimum base probability and the capped inactivity bonus ensure that no node is permanently excluded or permanently advantaged.

If no node proposes and none are forced, every node receives $0.1 \cdot R_{base}$ and the round is treated as a failure.

3.3 Learning and Equilibrium Analysis

Each node i trains a policy π_{θ_i} and a value function V_{ϕ_i} via independent Proximal Policy Optimization [24]. We use independent rather than centralized-critic training (e.g., MAPPO [25]) so that no inter-node global information is required at execution time, preserving the framework's decentralization property; the trade-off is non-stationarity, discussed at the end of this section.

Algorithm 2 Proposer Selection from History

Require: Proposing nodes N_{prop}, step t, histories H
Ensure: Selected proposer p^* (or `None`)
if $N_{prop} = \emptyset$ **then return** `None`
end if
$probs \leftarrow [\,]$
for each $i \in N_{prop}$ **do**
  $p_{base} \leftarrow \max(0.1, H_i.total_reward/100)$
  $p_{idle} \leftarrow \min(2.0, H_i.inactivity_count/5)$
  $p_{force} \leftarrow 2.0$ if $H_i.forced_proposal$ else 1.0
  $probs$.APPEND$(p_{base} \cdot p_{idle} \cdot p_{force})$
end for
Normalize $probs$ (uniform if all zero)
return RANDOMCHOICE$(N_{prop}, probs)$

Algorithm 3 Game-Theoretic Consensus

Require: Node actions $\mathcal{A}$, step t, global state s_g
Ensure: Proposer p^*, rewards $\mathcal{R}$, success flag
Initialize $N_{vol}, N_{forced}, N_{verify}, N_{wait} \leftarrow \emptyset$
for $i = 1$ to n **do**
  if SHOULDFORCEPROPOSAL(i, t) **then**
    $N_{forced} \leftarrow N_{forced} \cup \{i\}$
  end if
end for
for $i = 1$ to n **do**
  if $i \in N_{forced}$ **then continue**
  else if $a_i = 0$ **then** $N_{vol} \leftarrow N_{vol} \cup \{i\}$
  else if $a_i = 1$ **then** $N_{verify} \leftarrow N_{verify} \cup \{i\}$
  else $N_{wait} \leftarrow N_{wait} \cup \{i\}$
  end if
end for
$\tau \leftarrow \max(2, \lfloor n/3 \rfloor)$
if $|N_{vol}| \geq \tau$ **then**
  $\mathcal{R} \leftarrow$ APPLYPENALTY$(\cdots)$; **return** (`None`, $\mathcal{R}$, `False`)
end if
$N_{all} \leftarrow N_{vol} \cup N_{forced}$
if $N_{all} = \emptyset$ **then**
  $\mathcal{R} \leftarrow \{0.1 \cdot R_{base}\}_{i=1}^{n}$; **return** (`None`, $\mathcal{R}$, `False`)
end if
$p^* \leftarrow$ SELECTPROPOSER(N_{all}, t)
$\mathcal{R} \leftarrow$ CALCULATEREWARDS$(p^*, N_{vol}, N_{forced}, N_{verify}, N_{wait})$
return $(p^*, \mathcal{R}$, `True`$)$

Independent PPO update.

At each learning trigger, node i samples a minibatch $\{(s, a, r, s', d)\}_{j=1}^{B}$ from its buffer $\mathcal{D}_i$. Advantages are computed with the current value function and normalized within the minibatch:

$$\hat{A}_j = (r_j + \gamma V_{\phi_i}(s'_j)(1 - d_j) - V_{\phi_i}(s_j)), \quad \tilde{A}_j = \frac{\hat{A}_j - \mu_A}{\sigma_A + 10^{-8}}. \tag{4}$$

The old log-probabilities are *cached* from the parameters θ_i^{old} at the start of the update and held fixed for K_{epoch} inner epochs:

$$\ell_j^{\text{old}} = \log \pi_{\theta_i^{\text{old}}}(a_j \mid s_j) \quad \text{(no gradient)}. \tag{5}$$

Algorithm 4 Independent PPO Update for Node Agent i

Require: Buffer $\mathcal{D}$, minibatch size B, inner epochs K_{epoch}, clip ϵ, discount γ
Ensure: Updated $\pi_{\theta_i}, V_{\phi_i}$

if $|\mathcal{D}| < B$ **then return**
end if
$\mathcal{B} \leftarrow \textsc{SampleBatch}(\mathcal{D}, B)$; extract $\{(s_j, a_j, r_j, s'_j, d_j)\}$
$V_j \leftarrow V_{\phi_i}(s_j)$; $V'_j \leftarrow V_{\phi_i}(s'_j)$
$y_j \leftarrow r_j + \gamma V'_j(1 - d_j)$; $\hat{A}_j \leftarrow y_j - V_j$
$\tilde{A}_j \leftarrow (\hat{A}_j - \mu_A)/(\sigma_A + 10^{-8})$ ▷ normalize
$\ell_j^{\text{old}} \leftarrow \log \pi_{\theta_i}(a_j \mid s_j)$ with `no_grad` ▷ cache before update
for epoch $= 1$ to K_{epoch} **do**
    $\rho_j \leftarrow \exp(\log \pi_{\theta_i}(a_j \mid s_j) - \ell_j^{\text{old}})$
    $\mathcal{L}_{\text{policy}} \leftarrow -\frac{1}{B}\sum_j \min(\rho_j \tilde{A}_j,\ \text{clip}(\rho_j, 1-\epsilon, 1+\epsilon)\tilde{A}_j)$
    $\mathcal{L}_{\text{value}} \leftarrow \frac{1}{B}\sum_j (V_{\phi_i}(s_j) - y_j)^2$
    Update θ_i, ϕ_i with gradient-norm clip 0.5
end for
return $\pi_{\theta_i}, V_{\phi_i}$

Within each inner epoch, the probability ratio is evaluated in log-space against the current parameters:

$$\rho_j(\theta_i) = \exp\big(\log \pi_{\theta_i}(a_j \mid s_j) - \ell_j^{\text{old}}\big), \tag{6}$$

and clipped to produce the PPO surrogate objective

$$\mathcal{L}_{\text{policy}}(\theta_i) = -\frac{1}{B}\sum_{j=1}^{B} \min\big(\rho_j(\theta_i)\tilde{A}_j,\ \text{clip}(\rho_j(\theta_i), 1-\epsilon, 1+\epsilon)\tilde{A}_j\big), \tag{7}$$

with $\epsilon = 0.2$. The value function is updated by minimizing $\mathcal{L}_{\text{value}}(\phi_i) = \frac{1}{B}\sum_j (V_{\phi_i}(s_j) - y_j)^2$, where $y_j = r_j + \gamma V_{\phi_i}(s'_j)(1 - d_j)$. Both updates apply global gradient-norm clipping at 0.5; the old log-probabilities are floored at -4 for numerical stability, bounding the achievable ratio in early epochs. Caching ℓ_j^{old} before the update is essential: without it, the "old" and "current" distributions coincide and the ratio collapses to unity, reducing PPO to vanilla policy gradient. We make this implementation detail explicit because it materially affects training dynamics.

Stochastic-game formulation.

The system can be cast as a stochastic game [35] $\mathcal{G} = \langle N, S, \{A_i\}, P, \{R_i\}, \gamma\rangle$ with continuous state space $S \subset \mathbb{R}^d$ and finite per-node action set $A_i = \{0, 1, 2\}$. A stationary policy for node i is $\pi_i : S \to \Delta(A_i)$, with joint policy space $\Pi = \prod_i \Pi_i$ and discounted value function $V_i^{\pi}(s) = \mathbb{E}\big[\sum_{t\geq 0} \gamma^t R_i(s_t, a_t) \,\big|\, s_0 = s, \pi\big]$. We make the following standing assumptions:

(A1) *Boundedness:* $|R_i(s, a)| \leq M$ for some $M > 0$.
(A2) *Continuity:* $P(s' \mid s, a)$, $R_i(s, a)$, and $\pi(a \mid s; \theta)$ are continuous in their arguments.
(A3) *Compactness:* S is a compact metric space and each A_i is finite.
(A4) *Discounting:* $\gamma \in [0, 1)$.

ε-approximate Nash existence.

For any $\varepsilon > 0$, the continuous state space admits a finite ε-net $S_\varepsilon \subset S$ with mapping $\phi_\varepsilon : S \to S_\varepsilon$ such that $d(s, \phi_\varepsilon(s)) < \varepsilon$ for all s. The induced finite game G_ε inherits compactness and continuity from (A1)–(A4).

Lemma 1 (Existence in the finite game)**.** *G_ε admits a Nash equilibrium $\pi^*_\varepsilon \in \Pi_\varepsilon$.*

Proof sketch. The policy space Π_ε is a finite-dimensional compact convex set; the best-response correspondence is non-empty (finite MDPs admit optimal policies), convex-valued (linearity of V_i^π in π_i), and upper semi-continuous (continuity of V implies a closed graph). Kakutani's fixed-point theorem yields a $\pi^*_\varepsilon \in \mathrm{BR}_\varepsilon(\pi^*_\varepsilon)$. □

Theorem 2 (ε-Nash in the original game)**.** *Under (A1)–(A4), the original game $\mathcal{G}$ admits an ε-approximate Nash equilibrium $\pi^* \in \Pi$ for every $\varepsilon > 0$, obtained as a limit point of $\{\pi^*_\varepsilon\}_{\varepsilon \to 0}$.*

Proof sketch. Compactness of Π under the weak topology (Alaoglu) yields a convergent subsequence with limit π^*. If π^* were not ε-Nash, a profitable deviation would persist for sufficiently small ε by (A2), contradicting Lemma 1. □

Convergence.

Lemma 1 and Theorem 2 establish the *existence* of approximate equilibria. They do *not* establish that independent PPO converges to such an equilibrium. From any one agent's perspective, the environment is non-stationary — other agents' policies update concurrently — which violates the stationary-environment assumptions under which single-agent policy gradient is known to converge, and is a long-recognized obstacle in MARL [26]. We therefore make no probability-one convergence claim; the policies we deploy are the empirical fixed points of independent PPO under our training schedule, and Section 4 reports their behaviour rather than asserting optimality. Adopting a centralized critic in the style of MAPPO [25] is a natural direction for tightening this gap and is left to future work.

3.4 Adaptive Parameter Control and Fork Resolution

This subsection describes two systems-level mechanisms: an adaptive controller that adjusts incentive parameters online, and a fork-resolution protocol that guarantees eventual consistency.

Adaptive controller.

The controller is the tuple $\langle \Theta, \Pi, \Xi, \Upsilon \rangle$, with adjustable parameters $\Theta = \{R_{base}, P_{base}, \tau_{idle}, \delta\}$, performance metrics $\Pi = \{\theta_{tput}, f_{fork}, \rho_{success}, h_{network}\}$, an adaptation function Ξ, and a phase detector $\Upsilon(t) \in \{\textit{exploration}, \textit{exploitation}, \textit{stable}\}$. Every 10 steps, the controller computes normalized gaps $\Delta_m = (\bar{m} - m^*)/\max(\epsilon_m, m^*)$ for each metric (with $\epsilon_\theta = \epsilon_f = 0.05$, $\epsilon_\rho = 0.3$, clipped to $[-1, 1]$) and aggregates them

Algorithm 5 Fork Detection and Resolution

Require: Local chain B_{local}, finalized chain B_{global}
Ensure: Synchronized chain

if $\mathsf{hash}(B_{local}[-1]) \neq \mathsf{hash}(B_{global}[-1])$ **then**
  idx $\leftarrow -1$
  for $i = 0$ to $\min(|B_{local}|, |B_{global}|) - 1$ **do**
    if $\mathsf{hash}(B_{local}[i]) = \mathsf{hash}(B_{global}[i])$ **then**
      idx $\leftarrow i$; **break**
    end if
  end for
  $B_{local} \leftarrow B_{global}[0 : \text{idx} + 1]$
end if
if $|B_{local}| < |B_{global}|$ **then**
  $B_{local} \leftarrow B_{local} \cup B_{global}[|B_{local}| :]$
end if
ValidateChainContinuity(B_{local})
return B_{local}

as

$$\Delta^{\text{total}} = 0.4\,\Delta_\theta + 0.3\,\Delta_f + 0.3\,\Delta_\rho. \tag{8}$$

If $|\Delta^{\text{total}}| < \tau_{\text{stable}} = 0.1$, no adjustment is made. Otherwise an adjustment factor $\kappa = 0.1\,\Delta^{\text{total}}\,\lambda^{\text{phase}}$ scales four updates:

$$R_{base}^{(t+1)} = R_{base}^{(t)} \cdot \begin{cases} 1 + 0.5\kappa, & \text{early} \\ 1 + \kappa, & \text{otherwise} \end{cases} \tag{9}$$

applied when throughput falls below $0.2\theta^*$; $P_{base}^{(t+1)} = P_{base}^{(t)}(1 + 0.3|\kappa|\mu^{\text{phase}})$ when the fork rate is excessive; $\tau_{idle}^{(t+1)} = \tau_{idle}^{(t)}(1-0.3|\kappa|\xi^{\text{phase}})$ when network health $h = n_{active}/n$ falls below a phase-dependent threshold (0.6 early, 0.4 otherwise); and $\delta^{(t+1)} = \delta^{(t)}(1-0.2|\kappa|)$ in the late stage when $\Delta_\rho > 0.25$. Targets (θ^*, f^*, ρ^*) and the clamping bounds on Θ are set per environment. The controller's empirical behaviour, including the calibration of θ^* and a head-to-head comparison against fixed-parameter baselines, is the subject of Section 4.5; we do not claim adaptivity as an a priori advantage here.

Fork detection and resolution.

Each node i maintains a local replica B_{local}^i and compares the hash of its tip block to the hash of the latest globally finalized block: $\text{Fork}_i = \mathbb{1}[H_{local}^i \neq H_{global}]$. On detection, the node identifies the most recent common ancestor

$$\text{Ancestor}_{common} = \max\{k : H_{local}^k = H_{global}^k\}, \tag{10}$$

replaces the divergent suffix with the corresponding global segment, and validates chain continuity (the previous-hash field of every block must match the hash of its predecessor). To model realistic networks, node i receives the global consensus broadcast with probability $p_{recv}^i \in [0.5, 0.95]$, drawn from a five-category distribution at initialization. The protocol finalizes *at most one block per round*, which provides deterministic progression of the canonical chain; combined with the forced-proposer mechanism for inactive nodes, this guarantees liveness even under adversarial conditions. Algorithm 5 states the synchronization procedure.

3.5 Optional Components: Token Economy and Language-Model State Encoder

The framework also includes two auxiliary modules that we kept separable from the consensus core. We describe them briefly here and characterize their empirical contribution in Section 4.3; readers interested in implementation details are referred to the supplementary material.

Token-economy load generator.

Each node is paired with a *token agent* that performs small inter-agent transfers between consensus rounds: it inspects its balance, picks another token agent uniformly at random, and either initiates or requests a 10% transfer subject to balance thresholds. Successful transfers produce signed transaction records that are inserted into the corresponding node's transaction pool before the next consensus round. Operationally, the token system acts as an exogenous source of transaction load on top of the Poisson generator. The decision logic of the token agents is fully deterministic in the current implementation; integrating language-model-driven token decisions, as originally envisaged, is left for future work, particularly in settings where the inter-agent communication channel is itself semantic.

Language-model state encoder.

The local state $s_{base} \in \mathbb{R}^3$ may optionally be augmented with a semantic embedding produced by a pre-trained BERT encoder. A short textual description is generated from the global context (transaction-pool size, chain length, throughput, network health), with conditional clauses added when the pool exceeds 50 transactions or when the chain growth rate exceeds 0.1. The description is tokenized and passed through BERT; the final-layer CLS embedding $emb_{LM} \in \mathbb{R}^{d_{LM}}$ is concatenated with s_{base} to form the augmented state $s_{enhanced} \in \mathbb{R}^{3+d_{LM}}$. The augmented state, when used, replaces s_{base} as the policy network input.

Empirical scope of both modules.

The role of these modules in the framework is contingent on their measured contribution. We treat them as hypotheses to be tested rather than as established components: Section 4.3 reports a 2×2 factorial ablation that examines whether each module earns its computational and architectural cost on the consensus task. The conclusions drawn there determine the modules' role in the framework presented in this paper.

4 Performance Evaluation

We organize the evaluation around five research questions: **RQ1** — how does the framework perform and converge across network scales? **RQ2** — do the auxiliary components (LLM state enhancement, token-economy module) contribute measurable gains to the consensus core? **RQ3** — is the rapid policy-entropy collapse observed during training harmful, and does entropy regularization help? **RQ4** — does the adaptive parameter mechanism improve steady-state performance or robustness over

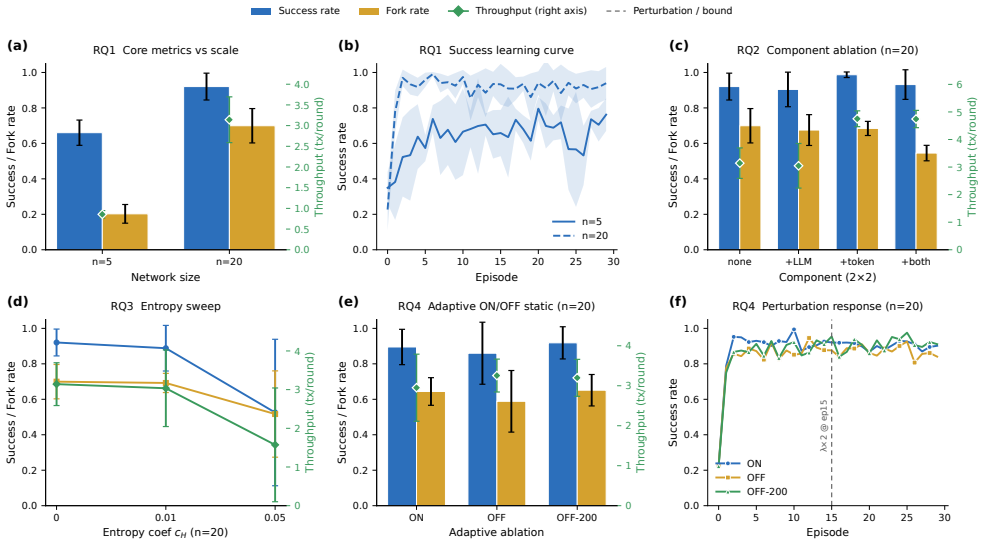


Fig. 3 Main evaluation results (5 seeds, mean ± std). (a) RQ1: core metrics across network scales; (b) RQ1: success-rate learning curves; (c) RQ2: 2×2 component ablation at n=20; (d) RQ3: entropy sweep — success rate, fork rate, and throughput vs. entropy coefficient; (e) RQ4: three-arm adaptive ablation (ON / OFF / OFF-200); (f) RQ4: performance under mid-training λ perturbation.

well-calibrated fixed parameters? **RQ5** — how does the system degrade under adversarial participation? Rather than reporting only favourable outcomes, we treat each design component as a hypothesis to be tested; where a component fails to earn its place, we say so and explain why. Figure 3 summarizes the main results.

4.1 Experimental Setup and Measurement Protocol

All experiments use the implementation described in Section 3: each node agent is trained with Proximal Policy Optimization with cached old-policy log-probabilities, clipped probability ratios (ϵ=0.2), per-minibatch advantage normalization, and global gradient-norm clipping (0.5). Unless otherwise stated, results are reported over **5 random seeds** as mean ± standard deviation, for networks of $n \in \{5, 20\}$ nodes trained for 30 episodes. Transactions are generated following Poisson(λ) and each broadcast transaction reaches a given node's pool with probability $p_{tx} = 0.8$. For RQ1–RQ4, all nodes receive consensus broadcasts with a uniform probability $p_{recv} = 0.95$; the adversarial experiments of RQ5 additionally model heterogeneous reception quality, with p_{recv} drawn uniformly from $\{0.95, 0.85, 0.75, 0.60, 0.50\}$ to reflect mixed network conditions.

Throughput metric.

Throughput is measured as the number of transactions confirmed *per consensus round* (tx/round). We deliberately avoid wall-clock-based throughput: in a simulated environment, wall-clock measurements conflate consensus behaviour with

Table 2 Core performance across network scales (mean ± std, 5 seeds, per-round throughput).

Metric	$n = 5$	$n = 20$
Throughput (tx/round)	0.86 ± 0.09	3.14 ± 0.55
Consensus success rate	0.660 ± 0.071	0.920 ± 0.076
Fork rate	0.203 ± 0.053	0.699 ± 0.097
Final chain length	39.4 ± 1.9	46.8 ± 4.9

the computational cost of the agents themselves (e.g., encoder inference), making results hardware-dependent and non-reproducible. The per-round metric is hardware-invariant and isolates the work actually performed by the consensus process. Complete hyperparameters and hardware specifications are provided in the reproducibility appendix.

4.2 RQ1: Performance and Convergence Across Scales

Table 2 summarizes steady-state performance, and Figure 3(a,b) shows the corresponding learning dynamics. Throughput scales with network size (0.86 → 3.14 tx/round from n=5 to n=20), as more participants both generate and confirm more transactions per round. Consensus success rate is higher at larger scale (0.92 vs. 0.66), reflecting that the congestion threshold $\gamma = \max(2, \lfloor n/3 \rfloor)$ is proportionally harder to violate in larger networks. Policies converge quickly: at n=20, runs sustain a success rate of 0.8 within 1–2 episodes.

Analysis of Fork Behaviour.

The fork rate rises markedly with scale (0.20 → 0.70). We verified that this is *not* an artefact of policy collapse or synchronized proposing: the number of concurrent voluntary proposers per step remains well below the congestion threshold (mean 3.2 at n=20; the probability of reaching the threshold is below 5%), and, as RQ3 shows, increasing policy stochasticity does not reduce forking without sacrificing success rate. Instead, forks at scale arise from the interaction of probabilistic broadcast reception with longest-chain synchronization: with more nodes, more local replicas can transiently diverge before the next finalized broadcast reconciles them. Because the protocol finalizes at most one block per round and resolves divergence from the last common ancestor (Section 3.4, Algorithm 5), these forks are transient by construction; they are the cost of autonomous (rather than leader-scheduled) block proposal, traded for decentralization.

4.3 RQ2: Do the Auxiliary Components Earn Their Cost?

A recurring concern with LLM-augmented systems is that language-model components are attached to, rather than integrated with, the decision core. We therefore treat both auxiliary components — the BERT-based state enhancement (Section 3.5) and the token-economy module (Section 3.5) — as hypotheses to be tested, via a full 2×2 factorial ablation (Table 3; Figure 3(c)).

Table 3 2×2 component ablation (mean ± std, 5 seeds). "Conv." is the first episode sustaining success rate ≥ 0.8; percentages give the fraction of seeds reaching it.

n	Config.	Success	Tx/round	Fork	Conv. (ep)
5	none	$0.660 \pm .071$	$0.86 \pm .09$	0.203	8.0 (100%)
5	+LLM	$0.668 \pm .048$	$0.90 \pm .10$	0.222	12.5 (80%)
5	+token	$0.741 \pm .082$	$1.44 \pm .50$	0.200	5.2
5	+both	$0.751 \pm .073$	$2.06 \pm .49$	0.213	9.2
20	none	$0.920 \pm .076$	$3.14 \pm .55$	0.699	1.6
20	+LLM	$0.904 \pm .098$	$3.05 \pm .81$	0.675	3.8
20	+token	$0.987 \pm .016$	$4.75 \pm .29$	0.684	1.6
20	+both	$0.932 \pm .084$	$4.74 \pm .31$	0.545	9.2

Effect of LLM-Based State Enhancement.

Across both scales, +LLM is statistically indistinguishable from the numeric-only baseline on success rate, throughput, and fork rate (all differences within one standard deviation), while each hypothesis under which semantic enhancement *should* help is individually falsified: convergence is slower (8.0 → 12.5 episodes at n=5, with only 80% of seeds reaching the threshold; 1.6 → 3.8 at n=20), early-phase performance is lower (0.766 → 0.647 mean success over the first five episodes at n=20), and run-to-run stability does not improve. Meanwhile the component inflates training wall-clock by 4–6× and the observation dimension by 768. The mechanistic reading is straightforward: the decision-relevant state consists of three low-dimensional numeric features over a three-action space; a 110M-parameter encoder embedding a textual rendering of those same numbers adds no information — only noise and latency. We therefore **demote the LLM enhancement to an optional module** and make no performance claim for it. We conjecture that semantic enhancement becomes valuable only when observations are irreducibly linguistic (e.g., natural-language proposals or inter-agent messages); we leave this to future work.

Effect of the Token-Economy Module.

The +token condition does improve success rate (0.920 → 0.987) and throughput (3.14 → 4.75) at n=20. Tracing the mechanism, however, shows that the token module acts as an *exogenous transaction source*: token transfers are injected into node transaction pools; fuller pools make block proposals both more likely to satisfy the non-empty-pool precondition and able to carry more transactions; both metrics rise accordingly. The improvement is therefore a property of increased transaction supply — which the consensus core absorbs well — rather than of agent-level economic behaviour, and we report it as a load-sensitivity observation rather than a contribution of the module. The only notable interaction effect, a lower fork rate for +both at n=20 (0.545 vs. 0.684), is accompanied by a 6× slowdown in convergence and lacks a mechanistic explanation; we do not consider it robust.

Table 4 Entropy sweep at n=20 (mean over 5 seeds). H is final-window policy entropy.

c_H	H	Fork	Success	Tx/round
0	0.005	0.699	0.920	3.14
0.01	0.028	0.692	0.888	3.04
0.05	0.090	0.517	0.525	1.57

4.4 RQ3: Entropy Collapse and Regularization

Independent-PPO training in our setting exhibits rapid policy-entropy collapse: from an initial entropy of ∼0.4 (action-space maximum $\ln 3 \approx 1.10$), policies become near-deterministic within 1–2 episodes ($H < 0.05$). A natural concern is that collapse traps agents in suboptimal deterministic strategies. We test this directly by sweeping the entropy-bonus coefficient $c_H \in \{0, 0.01, 0.05\}$ (Table 4; Figure 3(d)).

The entropy bonus does counteract collapse (H rises monotonically), and at c_H=0.05 the fork rate drops (0.70 → 0.52). This reduction, however, is not a repair but a Pareto-inferior trade: the same condition halves throughput (3.14 → 1.57) and collapses success rate (0.92 → 0.53), while the per-step count of concurrent voluntary proposers *increases* (mean 3.2 → 5.2; threshold-violation probability 0.05 → 0.43). Injected stochasticity suppresses successful block production overall, and forking falls only because fewer blocks exist to diverge. At n=5 the entropy–fork correlation is negligible ($r = -0.07$).

Impact of Entropy Collapse

The deterministic policies that collapse produces are not suboptimal in this task. The action space contains three options, the reward design makes role-appropriate actions strictly dominant in most states, and the forced-proposer mechanism guarantees liveness independently of policy exploration. Exploration therefore has little residual value once the small action ranking is learned, and near-deterministic policies achieve the best observed operating points. We conclude that entropy regularization is unnecessary in this environment — a boundary condition worth stating explicitly, since entropy bonuses are often applied by default in MARL systems.

4.5 RQ4: Is the Adaptive Parameter Mechanism Actually Adaptive?

The framework's original design premise is that online adaptation of incentive parameters improves performance under dynamic conditions. We subject this claim to a three-arm comparison: **ON** (the adaptive controller), **OFF** (parameters fixed at the controller's initial values: R_{base}=20, P_{base}=2, δ=0.8), and **OFF-200** (parameters fixed at the operating point the controller eventually reaches: R_{base}=200, P_{base}=3.0, δ=0.6). The third arm is essential: it distinguishes "the adaptation *process* adds value" from "the controller merely walks to a good fixed operating point." The controller's throughput target is calibrated to 1.15× the fixed-parameter steady-state throughput so that

Table 5 Adaptive-mechanism ablation at n=20, static regime (mean ± std, 5 seeds).

Arm	Success	Tx/round	Fork	Conv. (ep)
ON	$0.895 \pm .100$	2.95	0.644	1.6 (100%)
OFF	$0.860 \pm .175$	3.26	0.589	1.25 (80%)
OFF-200	$0.918 \pm .091$	3.20	0.651	4.0 (100%)

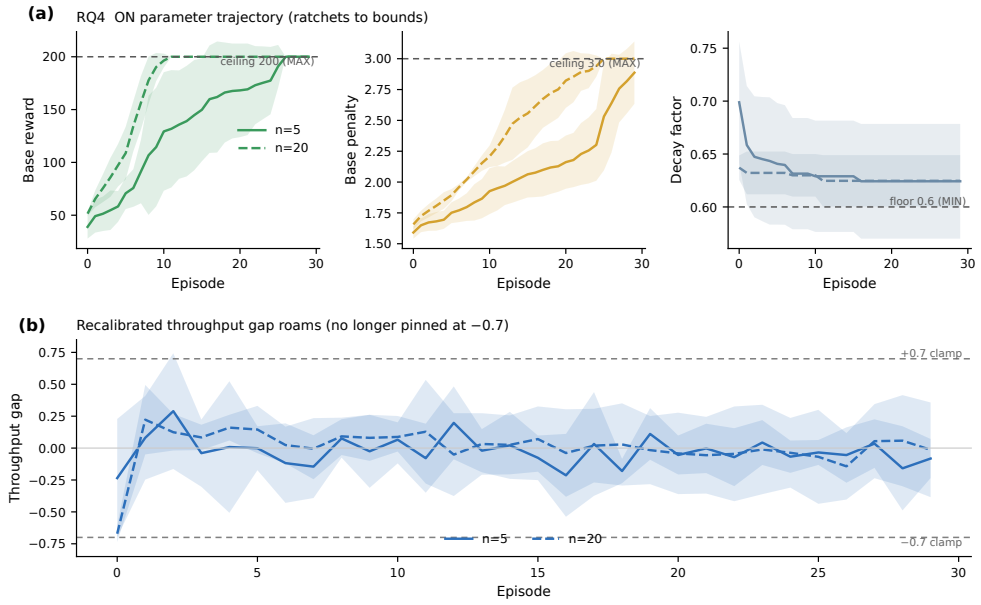


Fig. 4 Adaptive controller behaviour (ON arm, n=20). Top: base-reward, penalty, and decay-factor trajectories; the controller monotonically ratchets the base reward to its ceiling and pins the remaining parameters at their bounds, converging to the OFF-200 operating point. Bottom: throughput feedback gap after target calibration, wandering within $[-0.67, +0.29]$ rather than saturating.

its feedback signal is informative rather than saturated. We evaluate a static regime and a stress regime in which the transaction-generation rate λ is halved or doubled mid-training (Table 5; Figure 3(e,f); trajectories in Figure 4).

Performance of the Adaptive Controller.

In the static regime the three arms are statistically indistinguishable on success rate, throughput, and fork rate. Under λ perturbations, drops and recovery offsets are small for all arms ($|\Delta| \leq 0.07$) with no consistent ordering in ON's favour; the controller's protective adjustments trigger at most 0.4 times per run. Under the targets and bounds stated in Section 3.4, the adaptive mechanism improves neither steady-state performance nor robustness, and the natural hypothesis that dynamic load is its

advantage regime is not supported: well-calibrated fixed parameters absorb the same perturbations equally well.

Analysis of Adaptive Controller Dynamics.

Parameter trajectories (Figure 4) explain both the null result and ON's one consistent advantage. Although the calibrated throughput target keeps that feedback channel informative, the fork-rate and success-rate targets remain unattainable in this environment, so their channels monotonically ratchet the base reward to its ceiling ($20 \rightarrow 200$) and pin the penalty and decay factor at their bounds. The controller thus converges to precisely the fixed operating point from which OFF-200 starts — which is why their final performance coincides. ON's genuine advantage — faster, more reliable convergence (100% of seeds vs. 80% for OFF) — is therefore equivalent to a *reward-annealing schedule*: gradually raising the base reward early in training stabilizes learning, and a hand-designed warmup schedule would capture the same benefit without the adaptive machinery.

Scope of the claim.

This negative result is conditioned on the stated target values and clamping bounds; a controller with fully recalibrated, attainable targets on all channels might behave differently, and we flag this as an open question rather than concluding that adaptive incentive control is useless in general. Within the tested configuration, the summary is: *the framework's performance derives from its game-theoretic reward structure and the learned policies; the adaptive layer contributes a convergence-smoothing effect that a fixed schedule can reproduce.*

4.6 RQ5: Robustness Under Adversarial Participation

We evaluate adversarial robustness on a 20-node network across malicious fractions $\{10\%, 20\%, 30\%, 40\%, 50\%\}$, with 5 seeds per setting. Each malicious agent samples one strategy at initialization from a fixed set — network-latency injection, selfish withholding, double-spending (conflicting proposals), fork attempts, and passive non-participation — and executes it with probability 0.4 per step. The configuration is otherwise identical to RQ1: numeric-only state, the token-economy module disabled (to avoid confounding the degradation curve with exogenous load), and heterogeneous broadcast reception with p_{recv} drawn uniformly from $\{0.95, 0.85, 0.75, 0.60, 0.50\}$ across all nodes regardless of intent. Beyond success rate and per-round throughput, we report three indicators that capture how degradation manifests: *finality latency* (rounds from block proposal to global finalization), *fork-resolution time* (rounds from detection to synchronization), and *liveness* (finalized blocks per episode). Detailed numbers appear in Table 6; curves are in Figure 5.

Tolerance aligns with the protocol's $\lfloor n/3 \rfloor$ threshold.

Success rate is robust up to 20% malicious participants (≥ 0.83), exhibits a knee at 30% — precisely the $\lfloor n/3 \rfloor$ congestion threshold built into the consensus protocol (Section 3.2) — and collapses by 40%. This is the failure point the protocol's

Table 6 System behaviour vs. malicious fraction at n=20 (mean ± std, 5 seeds; finality, fork-resolution in rounds).

Mal. %	Success	Tx/round	Fork	Finality	Resol.	Liveness
10	0.931 ± .082	2.96	0.797	0.10	2.13	45.2
20	0.834 ± .092	2.68	0.808	0.16	1.67	43.3
30	0.671 ± .232	1.84	0.777	0.28	3.89	36.2
40	0.354 ± .287	0.81	0.605	0.15	8.10	22.1
50	0.316 ± .279	0.63	0.649	0.17	7.00	22.0

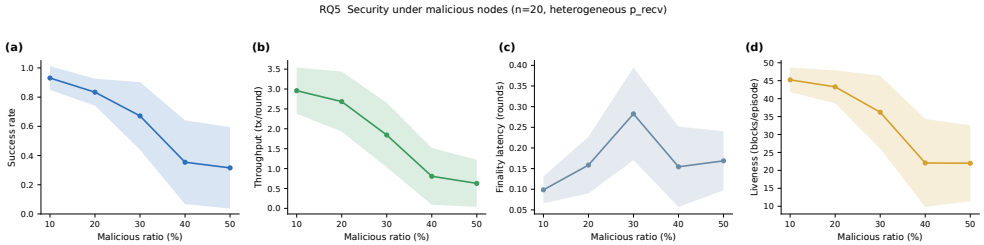


Fig. 5 Degradation under adversarial participation (n=20, 5 seeds). The cliff between 20% and 40% malicious participants brackets the protocol's $\lfloor n/3 \rfloor$ congestion threshold; finality stays low across the range, while liveness halves and fork-resolution time quadruples in the failure regime.

design predicts: once the count of voluntary (or here, deliberately disruptive) proposers exceeds $\lfloor n/3 \rfloor$, consensus rounds are penalized rather than completed, and the system's tolerance erodes accordingly. We make no claim of exceeding classical Byzantine fault tolerance bounds.

Decomposition of Performance Degradation.

The three new indicators reveal a structurally informative pattern that the success-rate column alone obscures. *Finality latency remains low across the full range* (0.10–0.28 rounds at every malicious fraction): blocks that do get finalized propagate quickly. What degrades is the production and reconciliation of blocks themselves: *liveness halves* from 45 to 22 blocks per episode between the safe and failure regimes, and *fork-resolution time quadruples* from $\sim$ 2 to $\sim$ 8 rounds. The summary is that surviving blocks are still finalized promptly, but fewer of them survive and divergences take markedly longer to heal. Reporting only success rate and throughput, as is customary, would have concealed this decomposition.

Run-to-run variance grows with adversity.

The standard deviation of success rate rises sharply in the failure regime (±0.23 at 30%, ±0.29 at 40–50%), reflecting a bimodal seed-level outcome: some seeds withstand the attack while others collapse.It indicates that average behaviour at the threshold should not be over-interpreted as a deterministic margin.

Table 7 Mixed-attack detection profile: per-node undetected-step counts (out of 1500 steps; smaller = better detected).

Attack	Undetected steps	Coverage
Passive non-participation	≈ 1500	none
Double spending	≈ 1500	none
Selfish mining	263	partial
Network latency	27	most
Explicit fork attempt	14	most

Sensitivity to Different Attack Types.

The framework operates without an explicit attack detector, relying on the consensus rules themselves to discipline misbehaviour. To characterize which attacks the rule set actually catches, we record per-step undetected-step counts under the mixed-attack runs (Table 7). Two strategies are essentially invisible to the current rule set: passive non-participation, which violates no protocol invariant and is only weakly penalized via the inactivity-bonus mechanism, and double-spending in the form of conflicting transaction submission, against which the consensus layer offers no direct check. Latency injection, selfish withholding, and explicit fork attempts, by contrast, are largely caught by the existing fork-detection and longest-chain reconciliation logic. These observations come from mixed-attack runs and should be read qualitatively; an isolated per-attack ablation (5 attacks $\times$ 5 ratios $\times$ 5 seeds $=$ 125 runs) is left to future work. The implication for design is direct: integrating a transaction-conflict checker and an activity-based slashing rule would close the two blind spots without altering the consensus core.

Architectural resistance, qualitatively.

Independently of the detection profile, two design choices contribute to the framework's behaviour under attack. Deterministic finalization of at most one block per round (Equation 2) eliminates the vulnerability window characteristic of probabilistic-finality protocols, which is consistent with the low finality latencies observed across all malicious fractions. Fork resolution against the globally finalized chain (Section 3.4) prevents persistent chain divergence even when local replicas transiently disagree, though it does not prevent the production of forks in the first place — and the rise in fork-resolution time at 40%+ malicious shows the cost when many divergences arise simultaneously. Identity binding, if needed in a deployment, would be supplied by an external mechanism (e.g., hardware attestation or certificate-based admission) rather than claimed as an experimental property of this framework.

4.7 Fairness of Reward Distribution

A consensus protocol whose rewards concentrate in a few nodes erodes its own decentralization: in this framework, accumulated reward feeds back into proposer-selection probability ($p_{base} = \max(0.1, h_i.\text{total_reward}/100)$ in Algorithm 2), so reward inequality compounds into participation inequality. We therefore examine the reward

Table 8 Reward-distribution fairness: per-node cumulative reward, post-convergence window vs. whole run (mean ± std, 5 seeds; all 5 seeds converged, 0 excluded).

	Post-convergence		Whole run	
n	G	CV	G	CV
5	0.034 ± 0.059	0.083 ± 0.147	0.035 ± 0.059	0.083 ± 0.145
20	0.045 ± 0.025	0.119 ± 0.064	0.045 ± 0.025	0.119 ± 0.064

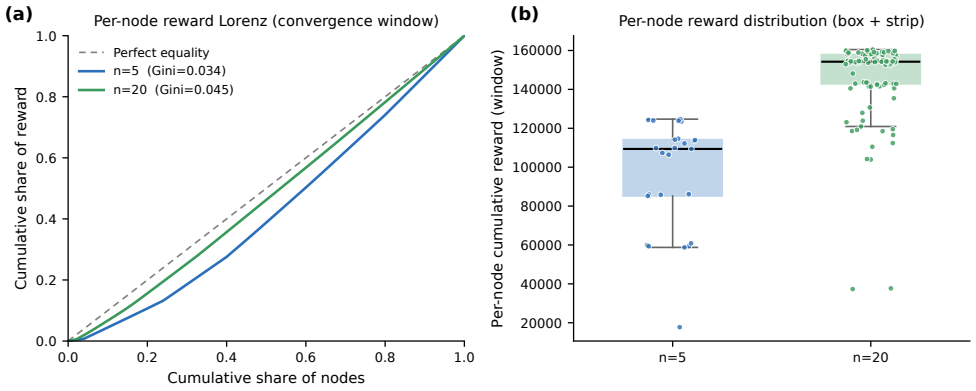


Fig. 6 Reward distribution within the post-convergence window. *Left:* Lorenz curves at n=5 and n=20, against the line of perfect equality. *Right:* per-node cumulative reward (box plot). Whole-run distributions exhibit a bimodal shape attributable to the exploration phase rather than cross-node inequality.

distribution induced by the game-theoretic mechanism, using the Gini coefficient G and the coefficient of variation CV as complementary measures of inequality and dispersion.

A reviewer of the previous version noted that the whole-run reward distribution is bimodal — a low-reward cluster and a concentrated high-reward cluster — which makes a single aggregate Gini value misleading. The bimodality is structural: the low cluster corresponds to the exploration phase before policies converge, and the high cluster to the post-convergence operating regime in which the fairness claim is meaningful. We therefore report fairness over a *post-convergence window*, defined per run as the interval following the first episode in which success rate reaches ≥ 0.8 (the same convergence criterion used in RQ2). Whole-run values are reported alongside for transparency.

Interpretation of the Fairness Results.

Table 8 and Figure 6 jointly address both the measurement question and the fairness claim.

(i) The bimodal whole-run distribution is a temporal artefact, not a cross-node inequality signal. The Gini coefficient is nearly identical between the post-convergence window and the whole run ($G = 0.034$ vs. 0.035 at n=5; 0.045 vs. 0.045 at n=20). This invariance has a direct explanation: the low-reward cluster in the whole-run histogram corresponds to the exploration phase, during which *all* nodes simultaneously earn low rewards. Removing that phase via the convergence-window filter changes every node's cumulative total by approximately the same amount, leaving the cross-node Gini unchanged. The bimodality is a timeline phenomenon, not a distributional one.

(ii) Per-node reward concentration is low throughout. Post-convergence Gini values of 0.034–0.045 indicate that no node accumulates a structurally dominant share; the Lorenz curves in Figure 6 lie close to the line of perfect equality. The coefficient of variation ($CV = 0.08$–0.12) confirms tight dispersion. The larger seed-level standard deviation at n=5 (± 0.059) reflects one seed producing a single outlier-low node; this small-network effect does not persist at n=20 (± 0.025).

These results resolve the reviewer-identified text–figure contradiction in the prior version. Under the corrected per-node measurement, the fairness claim is both numerically supported and explained. We do not draw comparisons to external economic inequality benchmarks, which conflate simulated protocol rewards with real-world income distributions.

5 Conclusion

This paper studied consensus among self-interested *learning* agents, a setting that falls outside the equilibrium assumptions of existing blockchain protocols. We cast the problem as a Markov game played by node agents trained with independent Proximal Policy Optimization, coupled to a game-theoretically motivated reward mechanism that aligns individual incentives with role-appropriate behaviour, and built a simulation environment supporting up to 100 nodes with deterministic per-round finality and longest-chain fork resolution. Across five research questions, we found that policies converge quickly under a hardware-invariant per-round throughput measure, that rewards are equitably distributed across nodes after convergence, and that adversarial tolerance aligns with the protocol's $\lfloor n/3 \rfloor$ threshold — a decomposition into finality, liveness, and fork-resolution time exposing how degradation manifests in ways that conventional metrics would conceal. We also report, with explanations rather than concealment, three negative findings: a language-model state encoder yields no measurable gain over a numeric baseline while inflating training cost; the token-economy module's apparent benefit is fully accounted for by exogenous transaction-pool injection; and the adaptive parameter mechanism, evaluated against calibrated fixed baselines, does not improve steady-state performance or robustness, with its sole consistent benefit equivalent to a reward-annealing schedule. We accordingly replace the framework's original probability-one convergence claim with an ε-approximate Nash existence result and an empirical account of what independent PPO settles to in practice, and identify centralized-critic variants such as MAPPO as a natural direction for tightening the non-stationarity gap. In sum, we present a framework that works in the regime we

tested, together with a record of the components that did not improve performance. We hope the latter is useful to subsequent work.

6 Discussion and Future Directions

The framework's contributions and limitations have a common source: small action spaces with strong dominance structure, combined with a forced-proposer liveness mechanism, make the consensus task amenable to lightweight learners and leave little residual room for auxiliary machinery to help. This explains both the positive findings of the previous section (independent PPO suffices; the game-theoretic reward design is the operative driver) and the negative ones (semantic state encoding, agent-level economic behaviour, and online parameter adaptation do not improve the consensus core in this regime).

Limitations.

We see three limitations that condition the scope of our claims, and that we did not paper over in the evaluation. *(i) Algorithmic.* We use independent PPO; the resulting non-stationarity from a single agent's perspective is acknowledged but not eliminated. Our equilibrium analysis establishes existence of ε-approximate Nash equilibria but does not guarantee that independent PPO converges to one, and the policies we report are empirical fixed points rather than certified optima. *(ii) Mechanistic.* The negative result for the adaptive controller (Section 4.5) is conditioned on the stated targets and clamping bounds: the fork-rate and success-rate targets remained unattainable in our environment, so two of the three feedback channels saturated. A controller with fully recalibrated, attainable targets on all channels might behave differently; whether such a recalibration recovers a meaningful adaptivity advantage is open. *(iii) Empirical.* Our evidence base is a simulation of up to 100 nodes with synthetic transaction load and a heterogeneous reception model; we make no claim about deployed-system performance, real network stacks, or implementation-level fault tolerance.

Future directions.

Four directions follow naturally from these limits. *First*, replacing independent PPO with a centralized-critic variant such as MAPPO [25] would tighten the gap between our existence result and what the learner actually attains, at the cost of requiring inter-node global information during training. *Second*, the negative result on LLM state enhancement is specific to settings where the observation is low-dimensional and numeric; semantic encoders may yet earn their cost in settings where the consensus content itself is linguistic. *Third*, the framework's tolerance profile aligns with the $\lfloor n/3 \rfloor$ congestion threshold (Section 4.6); raising this ceiling without abandoning autonomous proposal is a substantive open question, and integrating verifiable randomness or cryptographic admission as complementary mechanisms is one concrete avenue. *Finally*, validation beyond simulation — larger networks, real transaction streams, and deployment-grade tooling — would test whether the regime in which our findings hold persists outside the controlled environment we studied here.

Relation to Existing Consensus Protocols

The objective of this work differs from that of production blockchain protocols such as PBFT, HotStuff, Tendermint, Bitcoin, or Ethereum. Those systems are engineered for deployment under fixed protocol rules and are typically evaluated in terms of throughput, latency, and fault tolerance. By contrast, our focus is on consensus among learning agents whose policies evolve during operation.

Consequently, direct comparisons of simulated throughput or block production rates with deployed consensus protocols would not be informative, as the systems address different research questions. Rather than optimizing transaction-processing performance, this work examines whether consensus can be maintained when participant behaviour is non-stationary and incentives must adapt online.

From this perspective, the framework should be viewed as an experimental platform for studying consensus dynamics under learning participants rather than as a replacement for existing BFT, PoW, or PoS protocols. Future work could investigate whether the mechanisms developed here can be incorporated into more realistic consensus architectures and evaluated under production-oriented network models.

Acknowledgments

The author thanks his advisor for valuable guidance throughout this work, and his labmates for helpful discussions and feedback on earlier drafts.

Data Availability

The datasets generated and/or analysed during the current study are available from the corresponding author on reasonable request.

Declarations

Funding

The authors declare that no funds, grants, or other support were received during the preparation of this manuscript.

Conflict of Interest

The authors declare no competing interests.

References

[1] Berti, L., Giorgi, F., Kasneci, G.: Emergent Abilities in Large Language Models: A Survey. arXiv. arXiv:2503.05788 [cs] (2025). https://doi.org/10.48550/arXiv.2503.05788 . http://arxiv.org/abs/2503.05788 Accessed 2026-01-18

[2] Feng, T., Jin, C., Liu, J., Zhu, K., Tu, H., Cheng, Z., Lin, G., You, J.: How Far Are We From AGI. arXiv. arXiv:2405.10313 [cs] version: 1 (2024). https://doi.org/10.48550/arXiv.2405.10313 . http://arxiv.org/abs/2405.10313 Accessed 2026-01-18

[3] Zheng, Z., Xie, S., Dai, H., Chen, X., Wang, H.: An Overview of Blockchain Technology: Architecture, Consensus, and Future Trends. In: 2017 IEEE International Congress on Big Data (BigData Congress), pp. 557–564 (2017). https://doi.org/10.1109/BigDataCongress.2017.85

[4] Wang, W., Hoang, D.T., Hu, P., Xiong, Z., Niyato, D., Wang, P., Wen, Y., Kim, D.I.: A Survey on Consensus Mechanisms and Mining Strategy Management in Blockchain Networks. IEEE Access **7**, 22328–22370 (2019) https://doi.org/10.1109/ACCESS.2019.2896108 . Accessed 2026-01-18

[5] Eyal, I., Sirer, E.G.: Majority is not enough: bitcoin mining is vulnerable. Commun. ACM **61**(7), 95–102 (2018) https://doi.org/10.1145/3212998 . Accessed 2026-01-18

[6] Eyal, I., Gencer, A.E., Sirer, E.G., Renesse, R.V.: Bitcoin-NG: A Scalable Blockchain Protocol. In: 13th USENIX Symposium on Networked Systems Design and Implementation (NSDI 16), pp. 45–59. USENIX Association, Santa Clara, CA (2016). https://www.usenix.org/conference/nsdi16/technical-sessions/presentation/eyal

[7] Luu, L., Narayanan, V., Zheng, C., Baweja, K., Gilbert, S., Saxena, P.: A Secure Sharding Protocol For Open Blockchains. In: Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security. CCS '16, pp. 17–30. Association for Computing Machinery, New York, NY, USA (2016). https://doi.org/10.1145/2976749.2978389 . https://dl.acm.org/doi/10.1145/2976749.2978389 Accessed 2026-01-17

[8] Nakamoto, S.: Bitcoin: A Peer-to-Peer Electronic Cash System

[9] Bagaria, V., Kannan, S., Tse, D., Fanti, G., Viswanath, P.: Prism: Deconstructing the Blockchain to Approach Physical Limits. In: Proceedings of the 2019 ACM SIGSAC Conference on Computer and Communications Security. CCS '19, pp. 585–602. Association for Computing Machinery, New York, NY, USA (2019). https://doi.org/10.1145/3319535.3363213 . https://dl.acm.org/doi/10.1145/3319535.3363213 Accessed 2026-01-17

[10] Wood, D.G.: ETHEREUM: A SECURE DECENTRALISED GENERALISED TRANSACTION LEDGER

[11] Li, C., Li, P., Zhou, D., Yang, G.: A Decentralized Blockchain with High Throughput and Fast Confirmation

[12] Xiong, T., Xie, T., Xie, J., Luo, X.: ORIC: A Self-Adjusting Blockchain Protocol

with High Throughput. In: 2021 IEEE Intl Conf on Parallel & Distributed Processing with Applications, Big Data & Cloud Computing, Sustainable Computing & Communications, Social Computing & Networking (ISPA/BDCloud/SocialCom/SustainCom), pp. 1422–1434. IEEE, New York City, NY, USA (2021). https://doi.org/10.1109/ISPA-BDCloud-SocialCom-SustainCom52081.2021.00193 . https://ieeexplore.ieee.org/document/9644826/ Accessed 2026-01-18

[13] Martino, W., Popejoy, S.: Chainweb: A Proof-of-Work Parallel-Chain Architecture for Massive Throughput

[14] Yu, H., Nikolić, I., Hou, R., Saxena, P.: OHIE: Blockchain Scaling Made Simple. In: 2020 IEEE Symposium on Security and Privacy (SP), pp. 90–105 (2020). https://doi.org/10.1109/SP40000.2020.00008 . ISSN: 2375-1207. https://ieeexplore.ieee.org/document/9152798/ Accessed 2026-01-18

[15] Wang, J.: Monoxide: Scale out Blockchains with Asynchronous Consensus Zones

[16] Li, J., Xie, T., Wei, W., Xiong, T.: ORIC-Shard: A Scalable Blockchain Network with Sharding. In: Gao, H., Wang, X. (eds.) Collaborative Computing: Networking, Applications and Worksharing, pp. 23–42. Springer, Cham (2026)

[17] Buchman, E.: Tendermint: Byzantine fault tolerance in the age of blockchains. PhD thesis, University of Guelph (2016)

[18] Kokoris-Kogias, E., Jovanovic, P., Gasser, L., Gailly, N., Syta, E., Ford, B.: OmniLedger: A Secure, Scale-Out, Decentralized Ledger via Sharding. Publication info: Published elsewhere. 39th IEEE Symposium on Security and Privacy (2017). https://eprint.iacr.org/2017/406 Accessed 2026-01-18

[19] Zamani, M., Movahedi, M., Raykova, M.: RapidChain: Scaling Blockchain via Full Sharding. In: Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security. CCS '18, pp. 931–948. Association for Computing Machinery, New York, NY, USA (2018). https://doi.org/10.1145/3243734.3243853 . https://dl.acm.org/doi/10.1145/3243734.3243853 Accessed 2026-01-17

[20] Micali, S., Rabin, M., Vadhan, S.: Verifiable random functions. In: 40th Annual Symposium on Foundations of Computer Science (Cat. No.99CB37039), pp. 120–130. IEEE Comput. Soc, New York City, NY, USA (1999). https://doi.org/10.1109/SFFCS.1999.814584 . http://ieeexplore.ieee.org/document/814584/ Accessed 2026-01-18

[21] Tan, M.: Multi-Agent Reinforcement Learning: Independent vs. Cooperative Agents. In: Machine Learning Proceedings 1993, pp. 330–337. Elsevier, ??? (1993). https://doi.org/10.1016/B978-1-55860-307-3.50049-6 . https://linkinghub.elsevier.com/retrieve/pii/B9781558603073500496 Accessed 2026-01-18

[22] Lowe, R., WU, Y., Tamar, A., Harb, J., Pieter Abbeel, O., Mordatch, I.: Multi-Agent Actor-Critic for Mixed Cooperative-Competitive Environments. In: Advances in Neural Information Processing Systems, vol. 30. Curran Associates, Inc., ??? (2017). https://proceedings.neurips.cc/paper/2017/hash/68a9750337a418a86fe06c1991a1d64c-Abstract.html Accessed 2026-01-18

[23] Zhang, K., Yang, Z., Başar, T.: Multi-Agent Reinforcement Learning: A Selective Overview of Theories and Algorithms. arXiv. arXiv:1911.10635 [cs] (2021). https://doi.org/10.48550/arXiv.1911.10635 . http://arxiv.org/abs/1911.10635 Accessed 2026-01-18

[24] Schulman, J., Wolski, F., Dhariwal, P., Radford, A., Klimov, O.: Proximal Policy Optimization Algorithms. arXiv. arXiv:1707.06347 [cs] (2017). https://doi.org/10.48550/arXiv.1707.06347 . http://arxiv.org/abs/1707.06347 Accessed 2026-01-18

[25] Yu, C., Velu, A., Vinitsky, E., Gao, J., Wang, Y., Bayen, A., Wu, Y.: The surprising effectiveness of PPO in cooperative multi-agent games. Advances in Neural Information Processing Systems **35** (2022)

[26] Hernandez-Leal, P., Kaisers, M., Baarslag, T., Cote, E.M.: A survey of learning in multiagent environments: Dealing with non-stationarity. arXiv preprint arXiv:1707.09183 (2017)

[27] Mao, H., Netravali, R., Alizadeh, M.: Neural Adaptive Video Streaming with Pensieve. In: Proceedings of the Conference of the ACM Special Interest Group on Data Communication. SIGCOMM '17, pp. 197–210. Association for Computing Machinery, New York, NY, USA (2017). https://doi.org/10.1145/3098822.3098843 . https://dl.acm.org/doi/10.1145/3098822.3098843 Accessed 2026-01-17

[28] Mao, H., Schwarzkopf, M., Venkatakrishnan, S.B., Meng, Z., Alizadeh, M.: Learning scheduling algorithms for data processing clusters. In: Proceedings of the ACM Special Interest Group on Data Communication. SIGCOMM '19, pp. 270–288. Association for Computing Machinery, New York, NY, USA (2019). https://doi.org/10.1145/3341302.3342080 . https://dl.acm.org/doi/10.1145/3341302.3342080 Accessed 2026-01-17

[29] Naderializadeh, N., Sydir, J., Simsek, M., Nikopour, H.: Resource Management in Wireless Networks via Multi-Agent Deep Reinforcement Learning. IEEE Transactions on Wireless Communications **20**(6), 3507–3523 (2021) https://doi.org/10.1109/TWC.2021.3051163 . arXiv:2002.06215 [cs]. Accessed 2026-01-18

[30] Li, M., Long, X., Liu, M., Guo, J., Zhao, X., Wang, L., Wu, Q.: A decentralized adaptive method with consensus step for non-convex non-concave min-max optimization. Expert Systems with Applications **235**, 127159 (2025) https://doi.org/10.1016/j.eswa.2025.127159

[31] Shoham, Y.: Multiagent Systems: Algorithmic, Game-Theoretic, and Logical Foundations

[32] Saleh, F.: Blockchain Without Waste: Proof-of-Stake

[33] Kiayias, A., Russell, A., David, B., Oliynykov, R.: Ouroboros: A Provably Secure Proof-of-Stake Blockchain Protocol. Publication info: A major revision of an IACR publication in CRYPTO 2017 (2016). https://eprint.iacr.org/2016/889 Accessed 2026-01-18

[34] Zhang, J., Zhang, S., Zhang, G., Liao, G.: A data trading scheme based on blockchain and game theory in federated learning. Expert Systems with Applications **278**, 127158 (2025) https://doi.org/10.1016/j.eswa.2025.127158

[35] Shapley, L.S.: Stochastic games. Proceedings of the national academy of sciences **39**(10), 1095–1100 (1953)