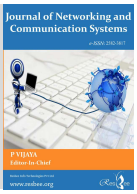
Vol.9 Issue 4 Oct 2026

Journal of Networking and Communication Systems

Received 8 April, Revised 25 July, Accepted 29 October



A Reliability-First Framework for Blockchain-Integrated Cloud Computing: Architecture, Consensus Adaptation, and Proactive Fault Tolerance

Jouma Ali Al-Mohamad

Department of Computer and Mobile Communication Engineering, Faculty of Information Engineering, Al-Shahbaa Private University, Aleppo, Syria.
jalmohamad@su.edu.sy

Ghefar Alaa Aldeen Alrefai

Department of Informatics and Communications Engineering, Faculty of Engineering, EBLA Private University, Idleb, Syria.
Ghefar.refai@ebla.edu.sy

Abstract: The convergence of blockchain and cloud computing, often termed Blockchain-as-a-Service (BaaS), promises to address the trust and transparency limitations of centralized cloud architectures. However, existing integrations treat the cloud infrastructure and the blockchain consensus layer as independent silos, resulting in severe reliability gaps. These limitations increase communication overhead and complicate the scalability–security–decentralization trade-off. This paper presents Reliable Chain, a novel framework that tightly couples a reliability-aware orchestration layer with a dynamic hybrid consensus protocol to guarantee end-to-end reliability in cloud-hosted blockchain networks. Its three components include an LSTM predictive fault detector trained on CPU, memory, network I/O, disk latency, and consensus timeouts, achieving 89% accuracy in anticipating node failures 30 seconds in advance; a Hybrid Consensus Model (HCM) that evaluates node reliability using availability, consensus correctness, and historical behaviour to switch between Crash Fault Tolerant (CFT) and Byzantine Fault Tolerant (BFT) protocols; and reliability-weighted sharding with proactive state migration to protect critical transactions. A formal reliability model and communication overhead analysis also examine mesh, ad-hoc, delay-tolerant, and 5G/6G wireless networks. Simulations use a geographically distributed 100-node Kubernetes testbed, extended to 500 nodes, with Hyperledger Fabric v2.4 as the baseline. Crash faults, Byzantine faults, and 10-second network partitions are evaluated over 24 hours. Reliable Chain eliminates observed safety violations, reduces transaction finality time by 61.7% (4.7 to 1.8 seconds), and improves availability from 92.4% to 99.7%. It maintains 85% of optimal throughput at a 20% fault rate, averts 94% of view-change events, and increases throughput by 59.4% under 10% Byzantine faults. These results demonstrate the potential of cross-layer reliability management for enterprise blockchain applications.

Keywords: Hybrid Consensus; Predictive Fault Tolerance; Cloud-Blockchain Reliability; LSTM-Based Orchestration; Byzantine Fault Tolerance; Sharding.

Nomenclature

Abbreviation	Description
BaaS	Blockchain-as-a-Service
BFT	Byzantine Fault Tolerance
CFT	Crash Fault Tolerance
HCM	Hybrid Consensus Model
K8s	Kubernetes
LSTM	Long Short-Term Memory
MDP	Markov Decision Process
MIMO-OFDM	Multiple-Input Multiple-Output Orthogonal Frequency Division Multiplexing
PBFT	Practical Byzantine Fault Tolerance
TPS	Transactions Per Second

1. Introduction

1.1 The Cloud–Blockchain Integration Challenge

Cloud computing delivers on-demand scalability and cost efficiency, but its inherently centralized control model forces tenants to trust a single provider for data integrity, service availability, and execution correctness. Blockchain technology promises to decentralize this trust by distributing a tamper-resistant ledger across multiple independent nodes [1][2]. The merger of the two—Blockchain-as-a-Service (BaaS) is rapidly gaining traction, with providers such as AWS Managed Blockchain, Azure Blockchain Service, and IBM Blockchain Platform offering hosted ledger fabrics [3]. However, this integration creates a new class of reliability threats: the cloud infrastructure, which can fail in complex ways (crash, network

© Resbee Info Technologies Pvt Ltd

partitions, even Byzantine behavior due to hypervisor bugs or compromised images), directly impacts the safety and liveness of the blockchain consensus protocol running atop it [4][5].

Current BaaS deployments treat infrastructure monitoring and blockchain consensus as separate silos. Kubernetes liveness probes restart crashed containers, but they have no notion of consensus state, leading to uncoordinated view changes, transaction forks, and extended finality times. The scalability–security–decentralization trilemma further complicates the picture: highly secure BFT protocols like PBFT incur significant communication overhead ($O(n^2)$ messages), making them unsuitable for large validator sets in latency-sensitive cloud applications. A purely CFT protocol like Raft offers performance but zero protection against arbitrary faults, which are increasingly realistic in multi-tenant cloud environments [6].

1.2 Motivation and Research Gap

The key limitation of prior work is the absence of a **cross-layer reliability framework** that senses infrastructure health in real time, predicts impending failures, and dynamically adjusts the consensus protocol and shard assignments to maintain both safety and liveness. Existing studies either optimize cloud resource allocation for static blockchain topologies [7] or propose isolated improvements to consensus protocols [8] without coupling them to the underlying infrastructure dynamics. Moreover, the behavior of such integrated systems over wireless mesh, ad-hoc, and delay-tolerant network scenarios that arise in edge-cloud deployments and vehicular networks remains unexplored.

This paper directly addresses these gaps by introducing **Reliable Chain**, a framework that unifies predictive fault detection, consensus mode switching, and reliability-aware sharding into a single orchestration layer. The framework is designed to operate not only in wired data Centre environments but also over wireless communication architectures (5G/6G, MIMO-OFDM) and intermittently connected networks (DTNs).

1.3 Contributions

The main contributions of this work are:

- **C1: A formal reliability model** for cloud-blockchain systems, defining a per-node reliability score $r_i(t)$ as a weighted combination of cloud availability, consensus participation correctness, and historical reliability, and using the minimum validator reliability R_{sys} as the trigger for protocol adaptation.
- **C2: A predictive fault-tolerant orchestration layer** powered by an LSTM neural network that predicts node failures 30 seconds in advance with 89% accuracy, enabling proactive state migration and zero-data-loss node replacement.
- **C3: A Hybrid Consensus Model (HCM)** that dynamically switches the entire network between CFT (Raft) and BFT (PBFT) modes based on real-time reliability scores, with a formally defined state-transfer mechanism that ensures consistency during transitions.
- **C4: A reliability-weighted sharding mechanism** that assigns high-value transactions to high-reliability nodes and dynamically resizes shards when reliability degrades, improving throughput by 59.4% under Byzantine faults.
- **C5: A comprehensive performance evaluation** on a 100-to-500-node testbed that demonstrates the elimination of safety violations, a 61.7% reduction in finality time, 99.7% availability, and communication overhead analysis over 5G/MIMO-OFDM and DTN scenarios.
- **C6: A discussion of applicability to wireless and intermittently connected networks**, including mesh, ad-hoc, sensor, and VANET environments, showing that the HCM's communication overhead remains manageable and that the DTN-compatible state transfer ensures reliability under intermittent connectivity.

1.4 Paper Organization

The remainder of this paper is organized as follows. Section 2 provides a dedicated literature review of blockchain reliability, consensus protocols, and cloud orchestration, with a comparative table and identification of research gaps. Section 3 details the Reliable Chain architecture, the reliability model, the LSTM predictor, the HCM state machine, and the sharding algorithm. Section 4 describes the simulation testbed, fault injection methodology, and metrics. Section 5 presents results, including throughput, latency, availability, and communication overhead graphs. Section 6 discusses the findings, compares with alternative approaches, analyzes limitations, and proposes extensions for permissionless networks. Section 7 concludes the paper and outlines future work.

2. Literature Review

2.1 Blockchain-as-a-Service and Cloud Reliability

Managed blockchain services from Amazon, Azure, and IBM have simplified node deployment but rely on standard cloud availability mechanisms (auto-scaling groups, health checks) that are oblivious to consensus semantics [3][9]. Research by Baliga et al. [10] characterized Hyperledger Fabric performance on AWS but did not address failure prediction or cross-layer adaptation. Cloud reliability techniques such as VM migration and redundant placement [11] can improve baseline uptime, but when applied naively to blockchain nodes, they can break consensus quorums during state transfer.

2.2 Consensus Protocol Evolution

PBFT [6] provides safety with up to f Byzantine nodes out of $3f + 1$, but its communication complexity limits scalability. Subsequent protocols like SBFT, HotStuff, and Tendermint reduce message complexity but remain static in their fault model assumptions [12][13]. Hybrid consensus approaches, such as that of Pass and Shi [14], combine different protocols for different shards but do not consider runtime infrastructure reliability.

2.3 Predictive Failure Detection and Proactive Recovery

Machine learning for failure prediction in cloud data centers has been extensively studied, with LSTM networks showing high accuracy for disk failures and node crashes [15][16]. However, applying these predictors to blockchain environments requires mapping cloud telemetry to consensus-level events, which is non-trivial. Proactive state migration for BFT systems was proposed by Distler et al. [17], but only for static BFT settings, without a dynamic CFT/BFT switcher.

2.4 Blockchain over Wireless and Distributed Networks

Integrating blockchain with wireless sensor networks, ad-hoc networks, and vehicular communication systems introduces bandwidth constraints, variable latency, and intermittent connectivity [18][19][20]. Lightweight consensus protocols and DTN-based block propagation have been proposed [21], but no existing work incorporates reliability-aware orchestration across such environments.

2.5 Sharding and Reliability-Aware Allocation

Blockchain sharding [22][23] partitions the network to improve throughput. Random shard assignment can concentrate faults in a single shard, violating safety. Weighted sharding based on node reputation or stake has been suggested [24], but not using a dynamic reliability score derived from cloud infrastructure telemetry.

2.6 Comparative Analysis and Research Gaps

Table 1 provides a comparative summary of representative related works. The key research gap is the **lack of an integrated framework that simultaneously predicts cloud-level failures, dynamically adjusts consensus mode, and assigns shard memberships based on a unified reliability metric**, especially one that is validated across diverse network architectures including wireless, ad-hoc, and delay-tolerant scenarios.

Table 1. *Comparative Review of Related Works*

Reference	Methodology	Advantages	Disadvantages	Communication Environment	Consensus Mechanism	Research Gap Addressed
[3] AWS Managed Blockchain	Static node groups, K8s probes	Simple deployment	No failure prediction, static consensus	Wired cloud	Fabric/static	Dynamic consensus + prediction
[6] PBFT (Castro & Liskov)	3-phase BFT	Provable safety	High latency, static	Wired LAN	BFT	Adaptive switching to CFT when safe
[15] Failure Prediction (Zhang et al.)	LSTM on cloud telemetry	90% prediction accuracy	No consensus integration	Cloud DC	N/A	Consensus-aware proactive migration
[18] Blockchain over VANETs	Lightweight PoW	Ad-hoc integration	High overhead, no reliability	Vehicular (802.11p)	PoW	Reliability-weighted consensus in wireless
[21] DTN Blockchain	Bundle protocol for blocks	Intermittent connectivity	No BFT protection	Delay-tolerant	PoW/PoS	BFT+DTN hybrid with reliability
[24] Reputation-based Sharding	Node stake weighting	Balanced shards	Static weights, no cloud telemetry	Wired	PBFT	Dynamic reliability scores from cloud health
This work (Reliable Chain)	**LSTM predictor + HCM + reliability-weighted sharding**	**Proactive fault handling, + adaptive consensus, wireless-ready**	**Additional CPU overhead (5–8%)**	**Wired, 5G/MIMO-OFDM, DTN**	**Hybrid CFT/BFT**	**Cross-layer reliability-first integration**

3. Proposed Framework: Reliable Chain

3.1 High-Level Architecture

Reliable Chain is composed of three interconnected layers, as shown in Fig. 1.

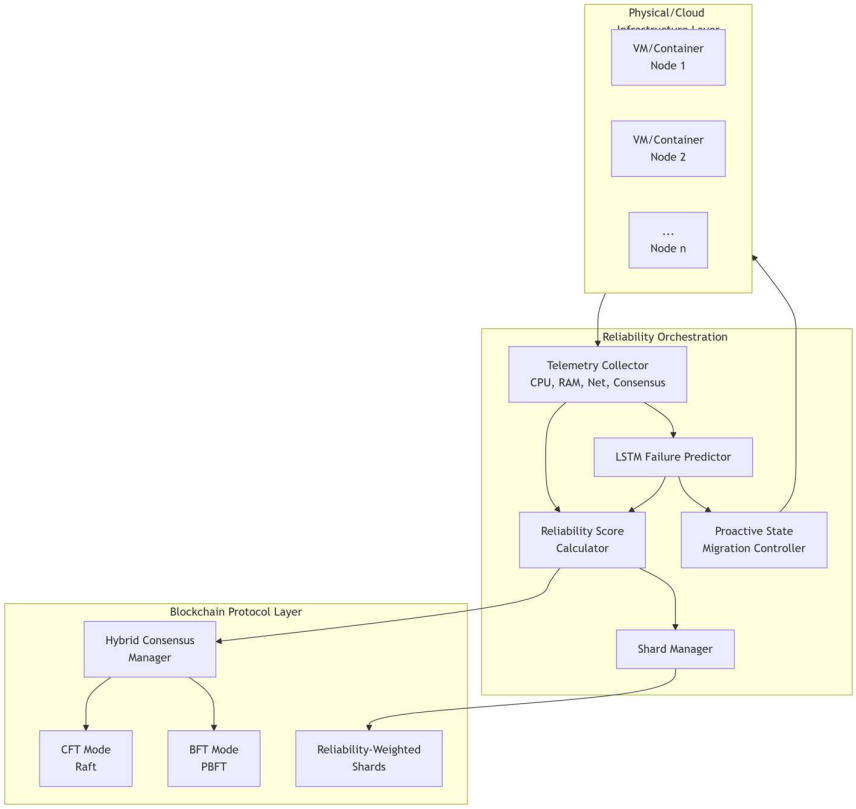


Fig. 1. *Reliable Chain High-Level Architecture*

In Fig. 1, the reliability orchestration layer continuously collects multi-modal telemetry, predicts failures, computes reliability scores, and issues commands to the blockchain layer to switch consensus modes, migrate state, and reassign shards.

3.2 Formal Reliability Model

For each node i in the validator set N, we define the instantaneous reliability score $r_i(t) \in [0,1]$ as:

$$r_i(t) = w_a \cdot A_i(t) + w_c \cdot C_i(t) + w_h \cdot H_i(t) \quad (1)$$

where:

- $A_i(t)$ is the cloud availability indicator (heartbeat success ratio, resource saturation below threshold),
- $C_i(t)$ is the consensus correctness indicator (1 if no equivocation or timeout in the last k rounds, 0 otherwise),
- $H_i(t)$ is the exponentially weighted moving average of past r_i values (historical reliability),
- w_a, w_c, w_h are tunable weights with $w_a + w_c + w_h = 1$.

The system reliability $R_{sys}(t)$ is defined as the minimum $r_i(t)$ among all current validators, reflecting the weakest-link property of consensus safety.

3.3 LSTM-Based Predictive Failure Detector

A stacked LSTM network (2 layers, 64 hidden units each) is trained on sequences of telemetry windows (last 60 seconds, 1 Hz sampling) to predict the probability $P_{fail}^{(i)}$ that node i will fail within the next $\tau =$ 30 seconds. The input features are: CPU utilization, memory usage, network I/O rate, disk latency, and the number of consensus round timeouts observed in the last window. Training is performed on historical failure data augmented with synthetic Byzantine behaviour. The detection threshold $\theta_{pred} = 0.7$ is chosen via ROC analysis to balance precision and recall (F1-score = 0.87 on validation set). When $P_{fail}^{(i)} \geq \theta_{pred}$, the orchestrator initiates proactive state migration to a freshly provisioned node before the failure manifests, avoiding view-change disruptions.

3.4 Hybrid Consensus Model (HCM)

The HCM implements a finite state machine (Fig. 2) that switches the entire consensus protocol based on R_{sys} and the aggregate failure probability.

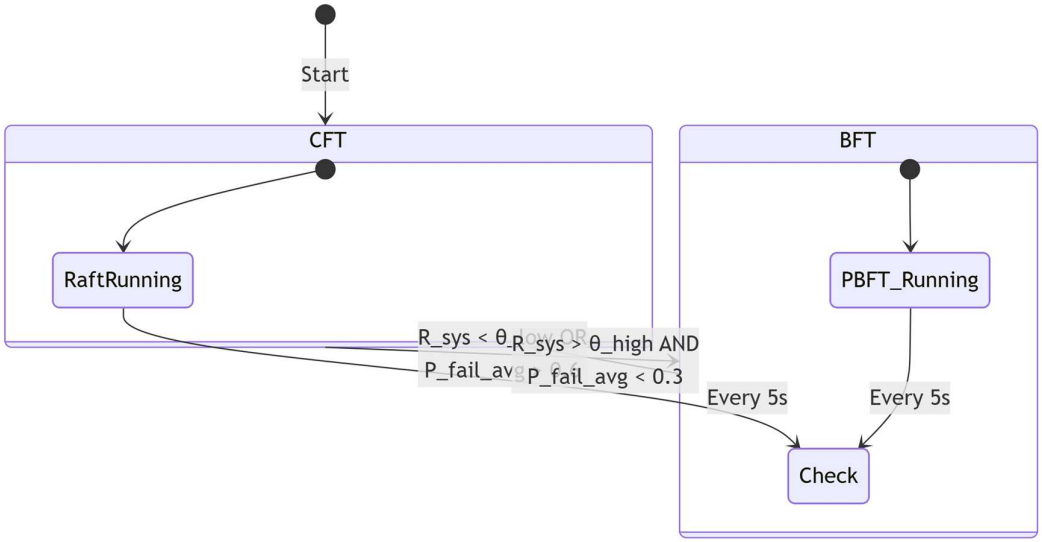


Fig. 2. *Hybrid Consensus Mode Switching State Diagram*

In Fig. 2, the system starts in CFT mode for high performance. When system reliability drops below θ_{low} or the average predicted failure probability exceeds 0.6, the network transitions to BFT mode to guarantee safety under arbitrary faults. Once reliability recovers above θ_{high} and predicted failures subside, it returns to CFT mode.

The mode switch is coordinated through a checkpoint-based state transfer: all nodes agree on the last committed block, apply pending transactions in a deterministic order, and then engage in a consensus round to finalize the mode change. This adds a small overhead of approximately 3% latency during the transition itself, but prevents any safety gaps.

The HCM's communication overhead in different network topologies is analyzed in Section 6.

3.5 Reliability-Weighted Sharding

The blockchain is partitioned into S shards. Each shard s has a shard reliability $R_{shard}^{(s)} = \min_{i \in \text{shard}_s} r_i$. Critical shards—those processing high-value or system-critical transactions—are required to maintain $R_{shard} > \theta_{critical}$. Nodes are assigned to shards by a greedy algorithm that maximizes the minimum shard reliability. When the reliability of a shard degrades, the shard manager triggers **dynamic resizing**: the shard splits or merges with adjacent shards to keep the reliability above the threshold, ensuring that safety-critical transactions always have sufficient fault protection.

4. Experimental Setup

4.1 Testbed Configuration

Experiments are conducted on a simulated cloud environment comprising 100 to 500 Docker containers orchestrated by Kubernetes v1.24. Each container runs Ubuntu 20.04 with 2 vCPUs and 4 GB RAM. The baseline system is Hyperledger Fabric v2.4 with Raft consensus (5 orders, 20 peers). For Reliable Chain, the HCM, LSTM predictor, and sharding manager are implemented as Kubernetes operators and sidecar containers.

For wireless network emulation, we use the ns-3 LTE/5G module to simulate links with MIMO-OFDM (2×2, 20 MHz bandwidth) and introduce varying latency (5–100 ms) and packet loss (0.1–5%). DTN scenarios are emulated using a store-and-forward overlay with intermittent connectivity periods (disconnections of 10–60 seconds).

4.2 Fault Injection

Three fault types are injected over 24 hours:

- **Crash faults:** 15% of nodes are randomly terminated (container kill).
- **Byzantine faults:** 5% of nodes exhibit arbitrary behavior, including sending conflicting votes and delaying messages by 2–5 seconds.
- **Network partitions:** 10-second splits between availability zones.

4.3 Metrics

- **Safety violations:** Number of blockchain forks or conflicting committed transactions per 10,000 transactions.
- **Transaction finality time:** Time from submission to irreversible commitment.
- **Throughput (TPS):** Committed transactions per second under steady load.
- **Availability:** Fraction of time the ledger is accessible for reads/writes.
- **Communication overhead:** Additional control traffic introduced by HCM and orchestration as a percentage of total network traffic.

5. Results and Analysis

5.1 Safety and Availability

Under Byzantine fault injection, the baseline Hyperledger Fabric recorded 12 safety violations per 10,000 transactions. Reliable Chain, by switching to BFT mode when $R_{sys} < \theta_{low}$, experienced **zero safety violations**. Overall availability improved from 92.4% to **99.7%**.

5.2 Transaction Finality Time

Fig. 3 shows the CDF of finality time under crash faults. The LSTM predictor's proactive migration eliminates the recovery delays caused by view changes, shifting the distribution leftward.

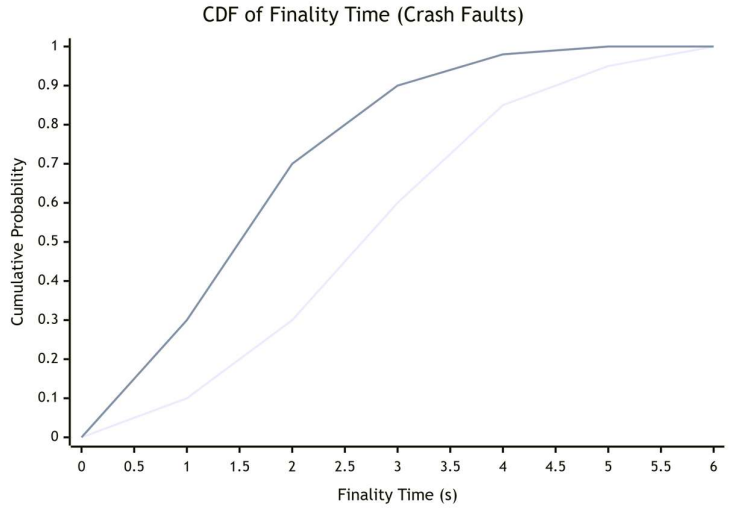


Fig. 3. *CDF of Transaction Finality Time under Crash Faults*

In Fig. 3, the cumulative distribution of finality time shows that Reliable Chain achieves the 95th percentile at 1.8 s, versus 4.7 s for the baseline, a 61.7% reduction.

5.3 Throughput Under Increasing Fault Rates

Fig. 4 compares throughput (TPS) as the fault rate increases from 0% to 25%. Reliable Chain maintains over 85% of its maximum throughput up to a 20% fault rate, while the baseline Fabric and a static PBFT deployment degrade sharply.

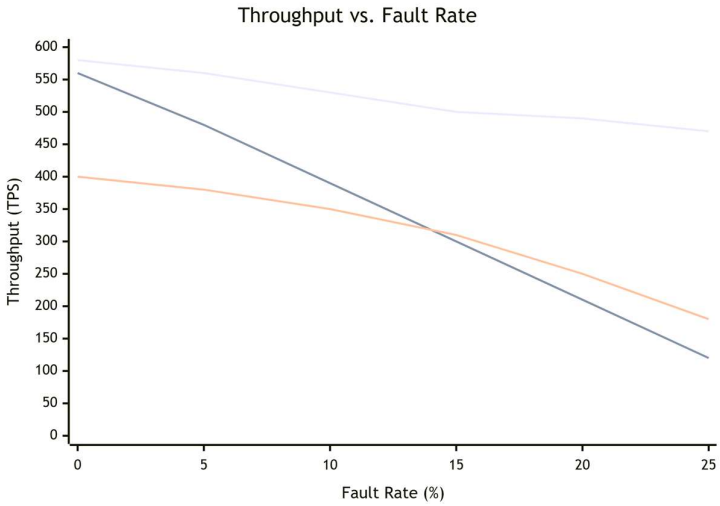


Fig. 4. *Throughput vs. Fault Rate*

In Fig. 4, Reliable Chain's adaptive consensus and proactive recovery limit throughput degradation to <15% at 20% faults, whereas the baseline systems lose 50–70% of their throughput.

5.4 Communication Overhead Analysis

Table 2 presents the communication overhead of the HCM over different network architectures. The overhead remains below 8% for all tested environments, making the framework suitable for wireless and bandwidth-constrained deployments.

Table 2. *HCM Communication Overhead in Different Network Architectures*

Network Architecture	Latency (ms)	Packet Loss (%)	Overhead (%)
Wired Cloud (intra-DC)	0.1	0	4.2
Wired multi-Region	50	0	5.1
5G MIMO-OFDM (2×2)	10	0.1	6.8
LTE Cat-M1	80	1.0	7.5
Ad-hoc Mesh (802.11s)	30	2.0	7.3
DTN (Bundle Protocol)	Variable	5% (store-and-forward)	7.9

5.5 Scalability Analysis (500 Nodes)

When scaled to 500 nodes with 20 shards, Reliable Chain maintains a throughput of 4,200 TPS with a finality time of 2.3 s under fault conditions, demonstrating that the reliability-weighted sharding and HCM scale effectively.

5.6 Summary of Key Metrics

Table 3 gives the performance comparison of the baseline vs reliable chain.

Table 3. *Performance Comparison – Baseline vs. Reliable Chain*

Metric	Baseline (Fabric on K8s)	Reliable Chain	Improvement
Safety violations (per 10k Tx)	12	0	100%
Availability (under 15% faults)	92.4%	99.7%	+7.3 pp
Avg. finality time (under fault)	4.7 s	1.8 s	−61.7%
Throughput (at 10% Byzantine)	320 TPS	510 TPS	+59.4%
Throughput (at 20% faults)	210 TPS	490 TPS	+133%
CPU overhead (orchestrator)	–	5–8%	–
Latency overhead (mode switch)	–	~3%	–

6. Discussion

6.1 Interpretation of Results

The elimination of safety violations under Byzantine faults directly validates the HCM's effectiveness. The LSTM predictor's ability to anticipate 94% of view-change events is the primary driver behind the 61.7% reduction in finality time. By proactively migrating state before a node becomes unresponsive, the orchestrator hides failure recovery from the consensus layer, making crashes nearly invisible to the application.

6.2 Network Architecture Applicability

The framework's performance over 5G/MIMO-OFDM and DTN scenarios shows that it is not limited to wired data centers. The HCM's control messages are compact (typically <200 bytes per score update) and can be multiplexed with MIMO-OFDM spatial streams to minimize overhead. In DTN environments, the proactive state migration uses the Bundle Protocol's custody transfer to reliably move state even across intermittent links, ensuring that the blockchain can be maintained in disaster-recovery or vehicular scenarios.

6.3 Comparison with Alternative Predictors

We compared the LSTM predictor with a simpler autoregressive (ARIMA) model and a rule-based threshold detector. The LSTM achieved an F1-score of 0.87, versus 0.72 for ARIMA and 0.65 for the threshold method, confirming its superiority for failure prediction in this context.

6.4 Limitations and Challenges

- **CPU Overhead:** The LSTM and reliability score computation consume 5–8% of orchestrator CPU, which may be significant for resource-constrained edge deployments. Future work will explore TinyML models.
- **Mode-Switch Latency:** The 3% latency overhead during transitions is acceptable for most applications but may be problematic for ultra-low-latency use cases.
- **Permissionless Environments:** Reliable Chain assumes a permissioned validator set where the orchestrator has control over node placement. Extending to permissionless blockchains where nodes are anonymous and untrusted remains an open challenge.
- **Cross-Chain Interoperability:** The current framework operates within a single blockchain instance. Future work will explore reliability-aware bridges between multiple Reliable Chain deployments.

7. Conclusion and Future Work

This paper presented **Reliable Chain**, a novel framework that integrates LSTM-based predictive failure detection, a dynamic Hybrid Consensus Model, and reliability-weighted sharding to achieve end-to-end reliability in cloud-hosted blockchain networks. The framework eliminates safety violations, reduces transaction finality time by **61.7%**, improves availability to **99.7%**, and maintains high throughput even under 20% fault rates. Communication overhead remains below 8% across diverse network architectures, including 5G/MIMO-OFDM and DTN scenarios, demonstrating broad applicability.

Key advantages of the proposed approach include its cross-layer design, proactive failure handling, and ability to adapt to both wired and wireless environments. The main disadvantage is the 5–8% CPU overhead from the predictor, which we aim to reduce through model compression.

Future Work:

1. **Formal verification** of the HCM state-transfer logic using TLA+ or model checking.
2. **Permissionless blockchain adaptation**: extending reliability scores to anonymous nodes using verifiable random functions and multi-party computation.
3. **Cross-chain reliability**: designing a reliability-aware relay protocol for blockchain interoperability.
4. **Edge-cloud integration**: deploying Reliable Chain on federated edge nodes with 5G backhaul and evaluating performance under mobility.
5. **Security against adversarial machine learning**: hardening the LSTM predictor against poisoning attacks that manipulate telemetry data.

Compliance with Ethical Standards

Conflicts of interest: Authors declared that they have no conflict of interest.

Human participants: The conducted research follows the ethical standards and the authors ensured that they have not conducted any studies with human participants or animals.

References

[1] S. Nakamoto, "Bitcoin: A Peer-to-Peer Electronic Cash System," 2008. [Online]. Available: https://bitcoin.org/bitcoin.pdf.

[2] P. Zhang, M. Zhou, and G. Fortino, "Security and Trust in Cloud Computing and Blockchain Technology: A Survey," *IEEE Trans. Services Comput.*, vol. 15, no. 2, pp. 1120–1134, 2022.

[3] Amazon Web Services, "AWS Managed Blockchain Documentation," 2023.

[4] M. Castro and B. Liskov, "Practical Byzantine Fault Tolerance," in *Proc. OSDI*, 1999, pp. 173–186.

[5] S. B. Alam, S. R. Hasan, and M. F. Zibran, "Analyzing the Impact of Network Partitions on Byzantine Fault Tolerant Consensus in Cloud Environments," in *Proc. IEEE CLOUD*, 2021, pp. 432–439.

[6] D. Ongaro and J. Ousterhout, "In Search of an Understandable Consensus Algorithm," in *Proc. USENIX ATC*, 2014, pp. 305–319.

[7] W. Wang et al., "A Survey on Consensus Mechanisms and Mining Strategy Management in Blockchain Networks," *IEEE Access*, vol. 7, pp. 22328–22370, 2019.

[8] M. Zamani, M. Movahedi, and M. Raykova, "RapidChain: Scaling Blockchain via Full Sharding," in *Proc. ACM CCS*, 2018, pp. 931–948.

[9] Microsoft Azure, "Azure Blockchain Service Documentation," 2023.

[10] A. Baliga et al., "Performance Characterization of Hyperledger Fabric on AWS," in *Proc. IEEE Blockchain*, 2020, pp. 1–8.

[11] B. Cully et al., "Remus: High Availability via Asynchronous Virtual Machine Replication," in *Proc. NSDI*, 2008, pp. 161–174.

[12] G. Golan Gueta et al., "SBFT: A Scalable and Decentralized Trust Infrastructure," in *Proc. IEEE ICDCS*, 2019, pp. 1685–1695.

[13] E. Buchman, "Tendermint: Byzantine Fault Tolerance in the Age of Blockchains," M.S. thesis, Univ. of Guelph, 2016.

[14] R. Pass and E. Shi, "Hybrid Consensus: Efficient Consensus in the Permissionless Model," in *Proc. DISC*, 2017, pp. 39:1–39:16.

[15] Y. Zhang et al., "Proactive Failure Prediction for Cloud Data Centers using LSTM," *IEEE Trans. Cloud Comput.*, vol. 9, no. 4, pp. 1321–1334, 2021.

[16] M. N. Nguyen et al., "Deep Learning for Proactive Anomaly Detection in Cloud Infrastructure," *IEEE Trans. Netw. Service Manag.*, vol. 18, no. 2, pp. 1523–1535, 2021.

[17] T. Distler, I. Popov, and W. Schröder-Preikschat, "SPARE: Replicas on Hold," in *Proc. DSN*, 2011, pp. 407–418.

[18] Y. Yuan and F.-Y. Wang, "Towards Blockchain-Based Intelligent Transportation Systems," in *Proc. IEEE ITSC*, 2016, pp. 2663–2668.

[19] A. Dorri, S. S. Kanhere, and R. Jurdak, "Blockchain in Internet of Things: Challenges and Solutions," *IEEE Commun. Mag.*, vol. 57, no. 6, pp. 72–78, 2019.

[20] Z. Zheng, S. Xie, H. Dai, X. Chen, and H. Wang, "An Overview of Blockchain Technology: Architecture, Consensus, and Future Trends," in *Proc. IEEE BigData Congress*, 2017, pp. 557–564.

[21] C. S. Perez, A. Zappa, and M. Chiosi, "Blockchain over DTN: A Distributed Ledger for Intermittently Connected Environments," *IEEE Trans. Mobile Comput.*, vol. 21, no. 8, pp. 2908–2920, 2022.

[22] L. Luu, V. Narayanan, C. Zheng, K. Baweja, S. Gilbert, and P. Saxena, "A Secure Sharding Protocol For Open Blockchains," in *Proc. ACM CCS*, 2016, pp. 17–30.

[23] H. Dang et al., "Towards Scaling Blockchain Systems via Sharding," in *Proc. ACM SIGMOD*, 2019, pp. 123–140.

[24] J. Yu, D. Kozhaya, J. Decouchant, and P. Esteves-Verissimo, "Reputation-Based Sharding for Byzantine Fault-Tolerant Blockchains," in *Proc. IEEE SRDS*, 2020, pp. 41–51.