MTC-SBC: Reputation-based Service Provision for Multi-tier Computing-enabled Sharded Blockchain

Md Monjurul Karim[a], Qiang Qu[a,*], Kashif Sharif[b], Muhammad Muzammal[c], Sujit Biswas[d]

[a]Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences, Shenzhen 518055, China
[b]School of Computer Science and Technology, Beijing Institute of Technology, Beijing 100081, China
[c]Department of Computer and Information Sciences, Northumbria University, Newcastle NE7 7XA, United Kingdom
[d]School of Science and Technology, City St George's, University of London, London EC1V0HB, United Kingdom

Abstract

The integration of Sharded Blockchain (SBC) into Multi-Tier Computing (MTC) often neglects heterogeneous resource capabilities across cloud, fog, and edge-tiers, leading to suboptimal Quality of Service (QoS) and inefficient cross-shard communication. To address these challenges, we propose MTC-SBC, a hybrid framework that synergizes subjective logic-based trust modeling with network slicing. Unlike traditional static approaches, our methodology dynamically assigns network slices to blockchain shards by coupling node reputation, derived from belief, disbelief, and uncertainty parameters, with computational resource states. We formulate this joint slice-placement and leader selection process as a Mixed-Integer Non-Linear Programming (MINLP) problem, aimed at minimizing processing delays and service costs while maximizing shard reliability. To ensure computational tractability at the edge, we employ a decomposition technique using Rotated Second-Order Cone (RSOC) relaxation and the Alternating Direction Method of Multipliers (ADMM). This hybrid approach ensures that shard allocation is optimized for resource efficiency and resilient against Sybil attacks. Extensive evaluations on a unified hybrid testbed demonstrate that MTC-SBC achieves a 25% throughput improvement and a 70% enhancement in task completion ratio over baseline protocols. Furthermore, the proposed resource optimization reduces energy consumption by approximately 34%, validating the framework's suitability for latency-critical and energy-constrained IoT applications.

Keywords: Blockchain; Multi-tier computing; Optimization; Reputation; Sharding

1. Introduction

Global mobile data traffic is projected to increase dramatically, rising from 109 exabytes (EB) per month in 2023 to 466 EB per month by 2030 [1]. This growth is fueled by the rapid adoption of Internet of Things (IoT) applications, including holographic communication, cyber-physical systems, e-health, augmented/virtual/mixed reality, connected autonomous vehicles (CAVs), unmanned aerial vehicle (UAV) swarms, and industrial robotics. These applications impose stringent requirements for high throughput, ultra-low latency, strong reliability, and seamless multi-device communication across diverse scenarios [2]. Addressing these requirements necessitates a synergy of advanced computing and networking paradigms rather than reliance on any single technology.

Multi-tier computing (MTC) forms the foundation by delivering low-latency communication, computation, and storage through distributed resources across cloud, fog, and edge-tiers [3, 4]. However, providing differentiated services to heterogeneous IoT applications further requires dynamic resource partitioning. Network slicing addresses this by creating independent logical networks over shared physical infrastructure, ensuring tailored Quality of Service (QoS) and key Performance Indicators (KPIs) for each service class [5]. While MTC ensures proximity-based computation and network slicing enables service isolation, both rely on secure coordination and trust across diverse stakeholders. Blockchain, particularly when enhanced with sharding, complements MTC and network slicing by providing a decentralized, scalable, and tamper-proof ledger for service orchestration. Through smart contracts, blockchain enforces automated agreements, ensures auditability, and establishes trust among untrusted parties [6, 7]. Among different blockchain types, consortium blockchains are especially well-suited for slicing-driven IoT environments, as they balance decentralization with administrative control. The joint integration of MTC, network slicing, and blockchain is therefore critical to address the scalability, latency, and security challenges of modern IoT-driven networks.

While the integration of MTC, network slicing, and blockchain offers a promising foundation for scalable and secure IoT service orchestration, realizing this vision in practice introduces several critical challenges. First, sharded blockchain systems often experience delayed transaction processing, particularly for real-time applications, as they prioritize throughput over latency [8]. In MTC environments,

*Corresponding Author: Qiang Qu
Email addresses: karim@siat.ac.cn (Md Monjurul Karim), qiang@siat.ac.cn (Qiang Qu), kashif@bit.edu.cn (Kashif Sharif), muhammad.muzammal@northumbria.ac.uk (Muhammad Muzammal), sujit.biswas@city.ac.uk (Sujit Biswas)

the varying computational capacities of edge, fog, and cloud tiers introduce additional latency, which current sharded systems fail to dynamically address [9]. Second, cross-shard communication in distributed MTC environments increases complexity and network congestion, especially when slicing isolates applications across different shards [10, 11]. This inefficiency hinders seamless interaction between applications. Third, QoS management remains a challenge, as sharding alone does not account for the diverse QoS requirements of network slices, leading to resource allocation inefficiencies [12]. Finally, ensuring security and trust in decentralized and heterogeneous environments is critical, as malicious nodes compromise transaction validity and system reliability [13].

On the other hand, high energy consumption in consensus mechanisms presents scalability concerns for resource-constrained edge devices. Existing cloud computing and multi-access edge computing (MEC) solutions are inadequate, with cloud computing requiring extensive data transfers over fronthaul lines [14, 15] and MEC facing limited computational resources and security vulnerabilities [16]. On top of that, the consensus mechanisms in MTC environments face further challenges, particularly when integrating sharded blockchains across heterogeneous tiers. These include maintaining scalability and ensuring interoperability between nodes with varying computational capacities. Moreover, managing shard overhead, dealing with fluctuating node availability, and minimizing network congestion are critical factors that affect consensus efficiency. Finally, dynamically computing reputation across tiers introduces additional complexity, requiring robust mechanisms to assess and maintain node trustworthiness. Collectively, these challenges underscore the need for an effective consensus protocol that ensures scalability, fault tolerance, and resilience under dynamic real-world conditions.

Several approaches have been proposed to address these challenges. However, they still exhibit notable limitations. For example, reputation-based blockchain systems enhance trust by using historical behavior for consensus and shard allocation [17]. These systems struggle with scalability in dynamic network slicing environments and often overlook real-time latency requirements. Moreover, sharding-based blockchain solutions improve scalability by partitioning the blockchain into parallel shards [18]. Still, they face challenges with cross-shard communication overhead and fail to dynamically adapt to latency variations across heterogeneous computing tiers. Slicing-based blockchain solutions leverage blockchain for decentralized resource allocation [19, 20]. Yet these solutions often rely on centralized control, which creates bottlenecks and reduces flexibility. They also do not fully address the QoS demands of diverse applications. In addition, many existing methods suffer from high energy consumption in consensus mechanisms and provide limited support for heterogeneous environments. As a result, their applicability to large-scale IoT networks remains constrained.

To address these limitations, we propose MTC-SBC, a novel sharded blockchain framework specifically designed for MTC ecosystems. The framework employs a three-tier architecture that strategically integrates edge, fog, and cloud resources to leverage their complementary capabilities. MTC-SBC leverages geographic proximity and computational heterogeneity to dynamically map network slices to blockchain shards based on application-specific QoS requirements. MTC-SBC also incorporates a reputation-based shard allocation mechanism grounded in subjective logic theory to evaluate node trustworthiness across heterogeneous tiers, ensuring reliable transaction validation. Unlike existing approaches, MTC-SBC introduces a cross-tier optimization mechanism that jointly considers QoS constraints and reputation scores during shard assignment. For consensus management, the framework adapts the Byzantine Fault Tolerance (BFT)-Smart protocol to multi-tier environments, enabling robust performance across nodes with varying computational capacities. Furthermore, MTC-SBC formulates the resource allocation challenge as an MINLP optimization problem that jointly minimizes block verification delays and service provisioning costs while maximizing system-wide reputation scores. The problem is addressed through an epigraph-based reformulation and an ADMM-driven iterative method with binary recovery, ensuring efficient resource allocation across heterogeneous network tiers.

The proposed MTC-SBC framework delivers substantial performance improvements for blockchain-enabled IoT applications in dynamic MTC environments. Through intelligent resource allocation, the system reduces block verification delays by up to 50% and improves resource utilization compared to existing sharded blockchain solutions. The reputation-based recruitment mechanism enhances security by systematically prioritizing high-trust nodes, thereby mitigating risks from malicious participants and establishing cross-tier reliability. Additionally, the dynamic slice-to-shard mapping mechanism ensures robust QoS provisioning by enabling ultra-low latency for critical applications (e.g., e-health monitoring) while supporting high throughput for large-scale IoT deployments, all while maintaining strict slice isolation. Comprehensive experimental evaluations demonstrate that MTC-SBC significantly outperforms state-of-the-art solutions across key performance metrics, including transaction latency, resource utilization efficiency, security resilience, and task completion rates.

The primary contributions of this work are as follows:

- **Tier-aware shard assignment:** We introduce a shard-assignment model that couples slice QoS to committee load and tier capacities via cross-tier latency constraints, yielding a constraint structure not captured in prior sharding or MTC studies.
- **Cross-tier reputation with bounded influence:** We design a subjective-logic update with temporal decay and influence caps across tiers, integrate it into committee feasibility and the objective, and analyze basic properties (boundedness and monotonicity under stationary honest rates).
- **Joint MINLP formulation with RSOC epigraphs:** We formulate the joint slice-to-shard and resource allocation problem and derive rotated second-order cone epigraphs

that enable an efficient convexified subproblem within each ADMM iteration.

- **Problem-structured ADMM and rounding:** We develop an ADMM solver with closed-form updates for dual variables and projections onto the relaxed assignment polytope, followed by a min-cost-flow rounding that preserves feasibility; we report optimality gaps against exact solvers on small instances.
- **System realization:** We adapt BFT-Smart to heterogeneous tiers with tier-aware batching and block size, integrate shard assignment and cross-tier reputation into the transaction path, and evaluate on a multi-tier testbed under diverse load and trust scenarios.

The remainder of this paper is organized as follows. Section 2 reviews related work on blockchain systems, reputation mechanisms, sharding techniques, and network slicing approaches. Section 3 presents the proposed multi-tier architecture, while Section 4 describes the corresponding system model and formulates the joint optimization problem. Section 5 presents the ADMM-based solution methodology. Section 6 provides comprehensive performance evaluation through experimental and analytical validation. Finally, Section 7 concludes the paper and outlines future research directions.

2. Related Work

The integration of blockchain with MTC and network slicing is essential to address the scalability, latency, and security demands of modern IoT-driven networks. Existing solutions often face limitations in dynamic, heterogeneous environments. This section reviews related work in four categories: reputation, sharding, slicing, and other consensus solutions. We discuss their contributions, highlight their limitations, and position our proposed MTC-SBC framework as a novel solution for MTC-enabled sharded blockchain. Fig. 1(a) reports the node conflict ratio as the number of nodes per shard increases. Fig. 1(b) summarizes how reputation-based, sharding-based, and slicing-based schemes together with MTC-SBC, satisfy six key requirements: scalability, responsiveness, processing, delay, storage, and security. The scores on each axis utilize the features in Table 1 and the performance indicators in Section 6 to reflect the typical capability of each design family. Higher values denote stronger performance for scalability, security, processing, and responsiveness, whereas lower values (closer to the center) represent minimized overhead for delay and storage. MTC-SBC combines the scalability of sharding-based solutions, the responsiveness of slicing-based solutions, and the security of reputation-based schemes, leading to the balanced multi-dimensional profile shown in Fig. 1(b).

2.1. Reputation-based Blockchain Solutions

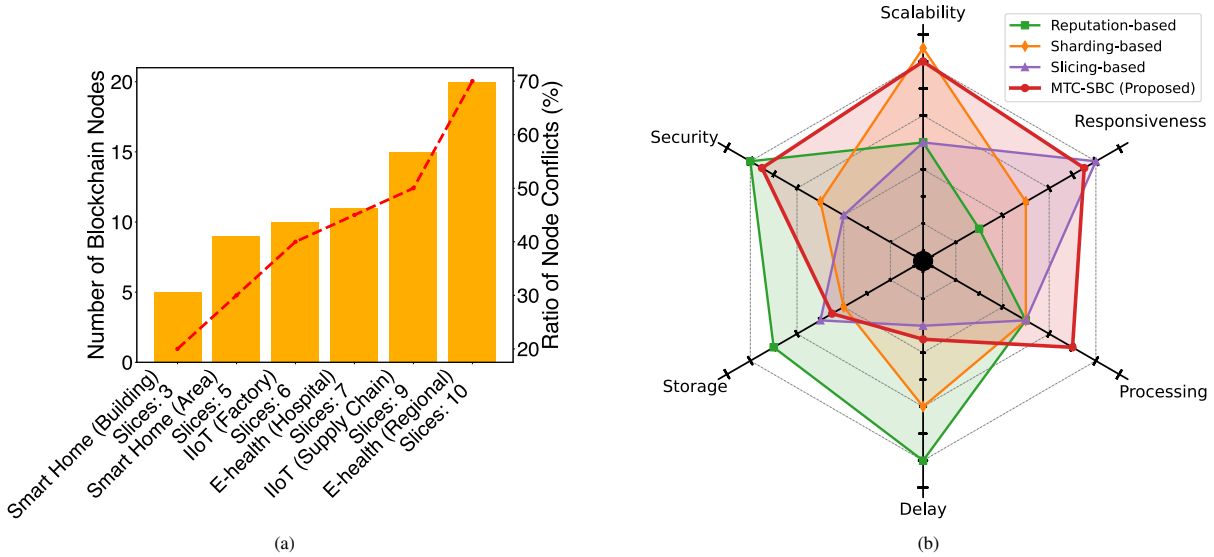
Reputation-based blockchain solutions [12, 21–31] leverage historical node behavior patterns to enhance trust establishment and consensus reliability in decentralized networks. Early approaches focus on replacing energy-intensive consensus mechanisms. For example, Proof of Reputation (PoR) [21] substitutes traditional mining with reputation-based block production to improve both security and energy efficiency. In IoT, reputation mechanisms enhance miner selection and mitigate collusion risks and develop global reputation scoring systems that preserve privacy while maintaining trustworthiness [22–25]. For industrial applications, several works [26, 27] propose dynamic reputation adjustment mechanisms specifically designed to secure Industrial IoT (IIoT) environments against Sybil attacks. Advanced reputation-based systems [12, 28] incorporate sharding and scalability optimizations to provide robust protection and improved security-throughput trade-offs. Recent studies [29–31] further integrate reputation-aware or blockchain-secured edge computing with complex task offloading and lightweight BFT consensus in MEC and IoT systems. Despite these advances, existing reputation-based solutions exhibit significant limitations for dynamic MTC environments. Most approaches do not capture cross-tier latency variations or slice-specific QoS constraints and do not jointly optimize shard formation, resource allocation, and consensus across edge, fog, and cloud tiers.

2.2. Sharding-based Blockchain Solutions

Sharding-based blockchain solutions [8, 10, 18, 32–39] partition the blockchain into smaller, parallel shards to enhance scalability. For instance, OmniLedger [32] achieves high scalability through atomic cross-shard transactions and bias-resistant validator assignment. Similarly, RapidChain [10] improves scalability by employing sub-linear communication complexity and resilient BFT for cross-shard transactions. Furthermore, Manshaei et al. [33] analyze strategic behaviors in shard-based blockchains to promote cooperation and address free-rider issues. Likewise, Monoxide [34] introduces asynchronous consensus zones to enable parallel transaction processing, while RepShard [8] integrates reputation-based sharding for secure shard formation. Moreover, Pyramid [35] proposes a layered sharding architecture to minimize cross-shard transaction overhead, and PolyShard [36] employs polynomially coded sharding to achieve linear scalability. Additionally, Set et al. [37] present a dynamic sharding approach tailored to service demands. In aggregate, these works face challenges such as cross-shard communication overhead and the absence of dynamic adjustments for MTC latency variations across tiers [38].

2.3. Blockchain-enabled Network Slicing

Blockchain-enabled network slicing solutions [11, 19, 20, 40–42, 47–55] use blockchain for decentralized resource allocation and management. For example, Kwantwi et al. [47] proposed a distributed blockchain-enabled network slicing framework for resource management. Zeydan et al. [48] introduced an architecture for multiple operators with a proof of business consensus protocol. Hewa et al. [49] provide secure, intelligent services in IIoT with federated slices based on tenant demands. Bukhari et al. [56] proposed a service-level agreements (SLAs) management trust



(a) (b)

Figure 1: Analytical comparison of (a) scalability challenges in terms of conflict ratio, and (b) functional trade-offs between MTC-SBC and existing solutions.

Table 1: Comparison between related works

Ref.	Consensus	Computing Layer	Reputation	Sharding	Slicing	Scalability	Overhead
[11]	BFT	Multi-domain	×	×	✓	Medium	High
[12]	BFT	Shard leader	✓	✓	×	Medium	Low
[20]	BFT	Core, Edge	×	×	✓	Medium	Medium
[23]	PoW	Cloud, Edge	✓	×	×	Medium	High
[24]	BFT	RSUs	✓	×	×	Medium	Medium
[34]	PoW	Cloud	×	✓	×	High	Medium
[35]	BFT	Peers	×	✓	×	Medium	High
[37]	PBFT	Peers	×	✓	×	Low	Medium
[39]	Multi-consensus	Cloud, Edge, End	✓	✓	×	High	High
[40]	Kafka, Raft	Cloud, Edge	×	✓	✓	High	High
[41]	PBFT	Cloud	✓	×	✓	Medium	High
[42]	PoW	Edge	×	×	✓	Medium	High
[43]	BFT	Edge	×	×	×	Medium	Low
[44]	PoS	Cloud, Edge	✓	×	×	Medium	Medium
[45]	BFT	Edge	✓	×	×	Medium	Medium
MTC-SBC	BFT-Smart [46]	Cloud, Fog, Edge	✓	✓	✓	High	Low

architecture using smart contracts. Boateng et al. [50] utilize blockchain-based deep reinforcement learning for resource trading among mobile virtual network operators. Moose [40] offers a scalable blockchain system for 5G applications, and NetChain [41] provides privacy-preserving multi-domain slice orchestration. Abdulqadder et al. [42] implement security and QoS provisioning in a 6G-MEC environment, while studies [51–55] explore blockchain-enabled network slicing brokering. These solutions often rely on centralized control, leading to bottlenecks, and do not fully address diverse QoS requirements in MTC environments.

2.4. *Other Consensus Solutions*

Other consensus solutions [43–45, 57] examine lightweight and federated learning (FL) mechanisms that aim to reduce consensus overhead and improve robustness in distributed systems. For example, Chacko et al. [57] survey several lightweight BFT protocols and group them by agreement method and round structure. The study reports design techniques that include reduced messaging complexity, committee sampling, and application-specific optimizations for IoT networks. These lightweight BFT-based schemes improve energy efficiency and latency. However, they primarily target single-tier infrastructures and do not account for cross-tier communication or computation delays, nor do they address slice-level QoS requirements. In parallel, recent FL-based blockchain solutions [43–45] secure model aggregation, support tamper-proof auditing of learning contributions, and incentivize reliable participants in MEC and edge environments. These studies show how consensus and ledger mechanisms strengthen decentralized learning in MEC-

IoT systems. Nevertheless, their focus remains on model training quality and incentive strategies instead of end-to-end transaction processing, shard formation, or slice placement in heterogeneous MTC environments.

2.5. Summary and Comparison

The reviewed works demonstrate advances in reputation, sharding, and slicing-based blockchain solutions, but they face significant limitations. Reputation-based solutions enhance security but lack scalability and low latency for network slicing. Sharding-based solutions achieve high throughput but struggle with cross-shard communication and MTC latency variations. Slicing-based solutions are limited by centralized control and inadequate QoS support. Figure 1a illustrates the increasing node conflict ratio as blockchain applications scale from small systems (e.g., smart homes) to large infrastructures (e.g., e-health networks), driven by inefficient node management and resource allocation. Our proposed MTC-SBC framework addresses these gaps by integrating edge, fog, and cloud computing with sharded blockchain and network slicing, using a reputation-based shard allocation model to optimize latency, enhance cross-shard communication, ensure QoS, and improve security. Figure 1b compares MTC-SBC with existing approaches across scalability, security, storage utilization, task processing, delay, and responsiveness, demonstrating its superior performance. Table 1 summarizes the comparison, highlighting MTC-SBC's high scalability, low latency, and robust QoS and security support.

3. Multi-tier Computing-enabled Sharded Blockchain System

In this section, we outline the proposed network architecture and explain its tiered structure. Besides, we introduce the blockchain framework and explain the core mechanisms and processes that ensure secure and efficient operations.

3.1. Proposed Network Architecture

Here, we present the architecture of our proposed system (shown in Fig. 2), which is divided into three distinct tiers (or layers):

- **Cloud-tier:** Cloud-tier includes highly-trusted resource-rich cloud processors (or data centers) maintained by the service provider that offers cloud-based solutions such as data analytics, content caching, resource management, computation offloading, security enforcement to the customers by leveraging the blockchain-as-a-service (BaaS) model. The peers located in the cloud tier have the highest authority, including decision-making and access to network information and other data. They possess sufficient processing and computational capabilities to perform hash calculations, digital signature verification, transaction validation, smart contract execution, and secure multi-party computation. The objective of these nodes is to improve overall network performance, including QoS maximization for all participants in the system.
- **Fog-tier:** The fog-tier consists of macro-cell base stations that assist lower level (i.e., edge-tier) nodes such as UAV swarms and other micro-cell base stations (BSs), respectively. Unlike the cloud-tier, the fog-tier nodes are usually low-trusted, have lower authority, lesser resource capabilities and are maintained by the virtual network operator (VNO). The objective of these nodes is to improve the performance of existing services by maximizing the QoS of the end-users and nodes located in the edge-tier.
- **Edge-tier:** The edge-tier comprises data-collecting units (DCUs) responsible for collecting data from end-users, such as mobile devices, terminals, people, and things. DCUs include roadside units (RSUs), micro-cell BSs, UAVs, ground control stations (GCSs), and Wi-Fi access points (APs), which are managed by the infrastructure provider (InP) and connected with other nodes in the edge, fog, and cloud-tiers through access, transport, and core routers, respectively. From a blockchain perspective, these nodes are typically untrusted, resource-constrained, privately owned, and have the lowest authority. Moreover, DCUs can also act as edge devices (EDs) to process data in proximity to end-users. In addition, DCUs are initialized with edge servers to perform necessary computational tasks upon receiving sensing data and task requests from end-users.

3.2. Control Plane and Service Orchestration

In the proposed MTC-SBC system, the data plane formed by cloud, fog, and edge nodes is coordinated by an SDN-based control plane that also interfaces with the blockchain layer. This subsection summarizes how controllers, shard committees, and network slices interact before we formalize the system model and optimization problem in Section 4.

Although the subjective-logic reputation update in Section 4.1 is applied uniformly to all computing nodes, the multi-tier architecture already encodes structural trust differences. Cloud-tier nodes are provisioned and controlled by the BaaS provider and operate in a more trusted administrative domain, whereas fog and edge nodes are owned by VNOs and infrastructure providers with higher exposure to adversarial behavior. The reputation mechanism therefore complements this tier-aware design by dynamically screening validators among fog and edge nodes where misbehavior is more likely, while leaving cloud-tier controllers as stable and highly trusted coordinators. To operationalize this hierarchical governance, we introduce distributed SDN controllers (e.g., root, domain) to ensure efficient resource orchestration across trusted and untrusted domains. These controllers are placed hierarchically in the cloud and fog-tiers to enable programmability and network management through softwarized orchestration. The root controller is responsible for coordinating services between different domain controllers by virtualizing slices on top of resources from the fog and edge-tiers. Consequently, its objective is to utilize the southbound

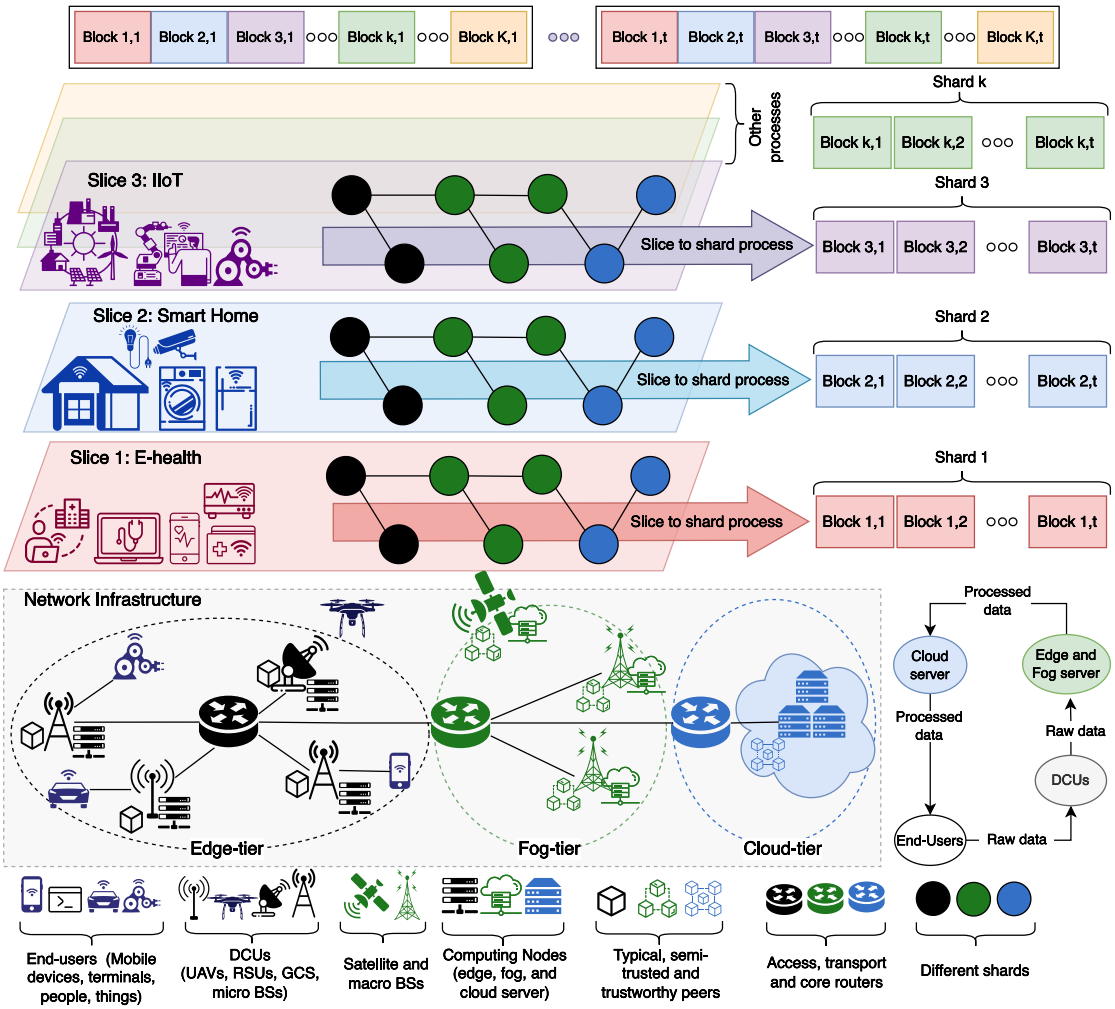


Figure 2: Proposed multi-tier computing enabled sharded blockchain architecture.

and northbound interfaces to identify logical resources through virtual slicing and match QoS/KPI requirements for each slice. Besides, these controllers are managed by the BSP, which acts as a regulator that constantly monitors cloud peers and recruits trustworthy peers among fog and edge nodes to ensure the security and reliability of the system. On the other hand, domain controllers are maintained by the VNO and perform resource virtualization by abstracting physical resources (e.g., radio resource blocks, power transmission) into logical resources, such as applications, radio bandwidth, and transmission rate. The primary responsibilities of the domain controllers include tracking traffic and bandwidth requirements of DCUs (or tenants) and forwarding analytics to the root controller. Lastly, tenants are individual users, companies, or organizations that subscribe to services provided by a VNO or a BSP and have specific service requirements regarding bandwidth, latency, security, and other performance metrics. Hence, the implementation of SDN controllers and network slicing in our system ensures that resources are optimally utilized to meet the specific requirements of each tenant while maintaining the security and efficiency of the system.

To optimize the interaction between shards and applications with QoS requirements, we allocate different applications (e.g., e-health, smart home, IIoT) to specific shards based on their frequency of interaction. These shards are organized within different slices, where each slice represents a distinct application domain [40, 58]. We use BFT-Smart for shard agreement on the state of the ledger. BFT-Smart is chosen over PBFT due to its superior performance and flexibility in real-world deployments. Unlike PBFT, which assumes a static replica configuration, BFT-Smart supports dynamic reconfiguration, allowing for seamless addition and removal of replicas without interrupting the system [46]. This flexibility is particularly valuable in MTC environments, where node availability fluctuates due to varying computational capacities across edge, fog, and cloud tiers. Moreover, BFT-Smart offers better scalability, enabling it to handle larger numbers of consensus participants without sacrificing fault tolerance or introducing excessive overhead, making it well-suited for heterogeneous MTC environments.

4. System Model and Problem Formulation

This section formalizes the MTC-SBC system and introduces the optimization problem that drives shard assignment

Table 2: Summary of important notations.

Symbols	Representation
$\mathcal{G} = \{\mathcal{N}, \mathcal{E}\}$	system graph with node set $\mathcal{N}$ and link set $\mathcal{E}$
$N,\ n,\ m$	total number of nodes; node indices
$N_{\mathbb{E}},\ N_{\mathbb{F}},\ N_{\mathbb{C}}$	node sets in edge, fog, and cloud tiers
$K,\ k,\ \lvert k\rvert,\ \chi_{n,k}$	number of shards; shard index; shard size; shard-assignment indicator
$\mathcal{J},\ j$	set of service types; service index
$\mathcal{J}_e^{\text{link}}(t)$	services using link e at timeslot t
$t \in \{0, \ldots, T\},\ \Delta t$	timeslot index; timeslot duration
$\mathbf{O}_{m\to n}$	subjective opinion from peer m to peer n (belief, disbelief, indecision)
$F^+_{m\to n},\ F^-_{m\to n}$	positive and negative feedback from peer m about peer n
$L_{m\to n}$	link quality between peers m and n
$\rho_n,\ \rho_k,\ \rho_k^{\text{total}}$	reputation of peer n; average and total reputation of shard k
$b_e(t),\ b_e^{\max}$	bandwidth of link e at timeslot t; maximum bandwidth of link e
$r_j,\ l_j,\ l_{j,k}^{\text{block}}$	data rate; data size; block data size for service j in shard k
$q_e(t),\ Q_n(t),\ L_{e,t},\ s_{e,t}$	virtual queue backlog; computation queue backlog; aggregate data volume and aggregate rate
$\phi_n(t),\ c_j$	computing capacity of node n at timeslot t; CPU cycles required by service j
$D_{j,k}^{\text{proc}},\ D_{j,k}^{\text{comp}},\ D_{j,k}^{\text{block}}$	processing, completion, and block verification delays
$\xi_{e,t},\ \Xi_e,\ \nu_x,\ \varpi_{e,t},\ \Psi_j,\ \eta_{n,k},\ \theta$	Lagrange multipliers

and resource allocation. We first describe the multi-tier network topology, reputation dynamics, and communication and computation processes. We then formulate a joint problem that couples end-to-end service delay, shard reputation, and service provisioning cost under capacity and consistency constraints. Figure 3 provides a high-level workflow that links these components to the consensus layer and highlights how the system evolves across timeslots.

The proposed system is modeled as an undirected graph $\mathcal{G} = \{\mathcal{N}, \mathcal{E}\}$, where $\mathcal{N} = \{1, \ldots, N\}$ is represented as the set of all participating nodes, i.e., peers and $\mathcal{E}$ is denoted as the links between these peers. Note that the participating nodes from edge, fog, and cloud-tiers denoted as $N_{\mathbb{E}}$, $N_{\mathbb{F}}$, and $N_{\mathbb{C}}$, respectively. These nodes are distinct and do not overlap such that $\mathcal{N} = N_{\mathbb{E}} \cup N_{\mathbb{F}} \cup N_{\mathbb{C}}$ and $N_{\mathbb{E}} \cap N_{\mathbb{F}} \cap N_{\mathbb{C}} = \emptyset$. Each node $n \in \mathcal{N}$ possesses heterogeneous computational capacity ϕ_n, reflecting real-world variations in processing power, memory, and network bandwidth across tiers. On the other hand, the communication link between a single peer m and peer n is defined as $e(m, n) \in \mathcal{E}$, where $m, n \in \mathcal{N}$. We segment the block verification process into discrete timeslots, represented as $t = 0, \ldots, T$, during which data (such as transactions) in successive blocks is processed, validated, and appended to the blockchain. Each timeslot has a duration equal to the block interval, or the phase time Δt. During a given timeslot, essential system characteristics, such as the number of blockchain nodes, transaction throughput, and computational capacity, remain constant but can vary in the following slot. Additionally, the system is divided into K shards, where each shard $k \in \mathcal{K} = \{1, \ldots, K\}$ consists of a group of nodes assigned to support specific network slices based on application requirements. A summary of the important notations is provided in Table 2.

4.1. Reputation-based Recruitment Model

To participate (or register) in the blockchain system, each peer $n \in \mathcal{N}$ submits its requests (e.g., IDs or digital signature) to the blockchain system for identification (and/or deposit, stake) that can be removed or withdrawn if they fail to follow specific system regulations, such as executing a specific task or validating a transaction in a timely manner, as set by the trustworthy peers or authoritative nodes, i.e., BaaS provider. In the practical blockchain system, it is difficult for the existing peers to acknowledge whether other peers are trustworthy or malicious. For example, any peer with malicious behavior generates incorrect transactions and false block verification results, letting the blockchain system reject the correct blocks while accepting the erroneous blocks [13]. Hence, we assign reputation to each peer in the blockchain system and use a subjective logic model to allow a peer N_n to update the reputation (i.e., opinion, feedback or trustworthiness) level of any other (or random) peer N_m by integrating its own feedback with other peers' opinion based on the past observations and interactions. The subjective opinion vector $\mathbf{O}_{m\to n} = \{\beta_{m\to n}, \delta_{m\to n}, I_{m\to n}\}$ is used to indicate the feedback of peer N_m regarding another peer N_n where $\beta_{m\to n}, \delta_{m\to n}$, and $I_{m\to n}$ represent belief, disbelief, and indecision [23]. Note that for all $m \in \{0\} \cup \mathcal{N}\backslash\{n\}$, the belief, disbelief, and indecision are such that $\beta_{m\to n} + \delta_{m\to n} + I_{m\to n} = 1$ and $\beta_{m\to n}, \delta_{m\to n}, I_{m\to n} \in [0, 1]$. Given that all peers have similar assessment criteria, the belief, disbelief and indecision are updated at the end of every timeslot t according to

$$\begin{cases} \beta_{m\to n} = (1 - I_{m\to n}) \frac{F^+_{m\to n}}{F^+_{m\to n} + F^-_{m\to n}} \\ \delta_{m\to n} = (1 - I_{m\to n}) \frac{F^-_{m\to n}}{F^+_{m\to n} + F^-_{m\to n}} \\ I_{m\to n} = 1 - L_{m\to n} \end{cases}, \tag{1}$$

where $F^+_{m\to n}$ and $F^-_{m\to n}$ are the positive and negative feedbacks assessed by peer N_m about peer N_n from the past observations

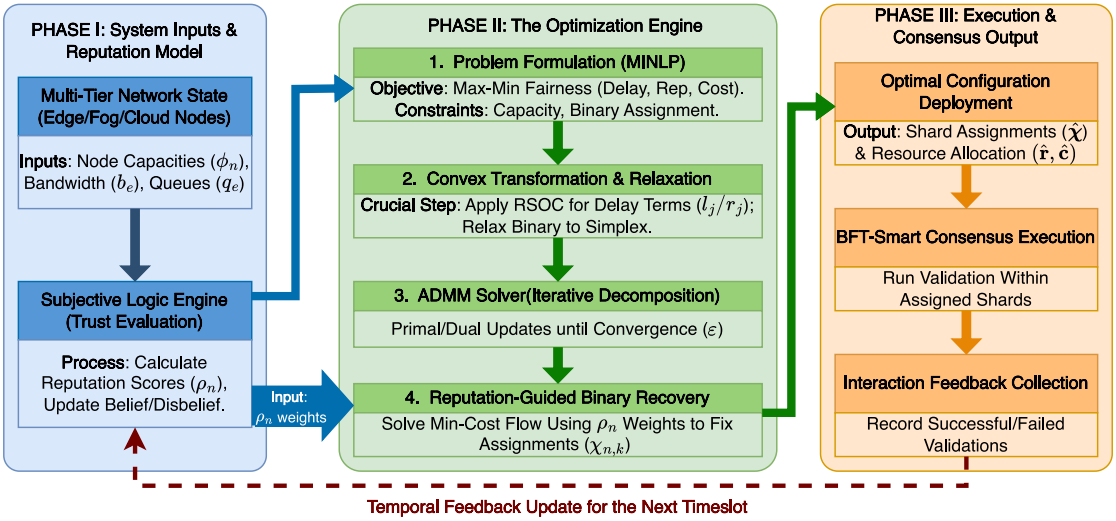


Figure 3: End-to-end workflow of the proposed MTC-SBC framework.

of interactions between N_m and N_n. In addition, $L_{m\to n}$ is the probability of successful states (or quality) of communication link between N_m, N_n and also the trustworthiness indicator of communication between N_m and N_n. At the beginning of system operation, for example when two peers have not yet interacted, no feedback is available and $F^+_{m\to n} + F^-_{m\to n} = 0$ holds. In this initial state, we adopt a neutral subjective opinion by setting the belief mass $\beta_{m\to n} = 0$, the disbelief mass $\delta_{m\to n} = 0$, and the uncertainty mass $I_{m\to n} = 1$. This configuration signifies full uncertainty and also avoids division by zero in Eq. (1). The resulting initial reputation contribution is $\rho_{m\to n} = \beta_{m\to n} + \mu I_{m\to n} = \mu$, which establishes that newly admitted nodes begin with a tier-agnostic prior trust level determined by μ. As positive and negative interactions accumulate, $F^+_{m\to n}$ and $F^-_{m\to n}$ become non-zero, the uncertainty $I_{m\to n}$ decreases, and the reputation gradually evolves from the neutral prior toward a belief or disbelief that reflects observed behavior. Therefore, the reputation $\rho_n(t)$ of peer N_n in timeslot t is presented as:

$$\rho_n(t) = \sum_{m\in\mathcal{N}\setminus\{n\}} \rho_{m\to n} = \sum_{m\in\mathcal{N}\setminus\{n\}} (\beta_{m\to n} + \mu I_{m\to n}), \quad \forall n \in \mathcal{N}. \quad (2)$$

Here, $\mu \in [0, 1]$ is a system-level hyper-parameter that controls how much the indecision term $I_{m\to n}$ contributes to the expected reputation. In the subjective logic framework, the expected probability that an entity is trustworthy is given by its belief plus an uncertainty term weighted by a base rate [23]. In our model, this corresponds to $\rho_{m\to n} = \beta_{m\to n} + \mu I_{m\to n}$, where μ plays the role of the base rate that captures the prior belief when the evidence is uncertain. Setting $\mu = 0$ reduces $\rho_{m\to n}$ to pure belief (indecisive evidence is ignored), while $\mu = 1$ treats fully undecided opinions as fully trusted, which is undesirable in adversarial environments. Therefore, μ is selected within a moderate interval (e.g., $\mu = 0.5$ in our evaluation) to reflect a neutral and conservative prior, where undecided interactions are neither fully trusted nor fully discarded.

It is also important to note that μ does not arbitrarily change the qualitative behavior of the reputation system. From Eq. (2), the reputation of node n is expressed as

$$\rho_n = \sum_{m\in\mathcal{N}\setminus\{n\}} (\beta_{m\to n} + \mu I_{m\to n}) = \underbrace{\sum_{m\in\mathcal{N}\setminus\{n\}} \beta_{m\to n}}_{\text{evidence}} + \mu \underbrace{\sum_{m\in\mathcal{N}\setminus\{n\}} I_{m\to n}}_{\text{uncertainty term}}.$$

As nodes accumulate interaction history, the indecision $I_{m\to n}$ decreases and the belief component $\sum_m \beta_{m\to n}$ dominates ρ_n. Furthermore, shard formation and voting decisions depend on ρ_n only through relative comparisons, such as the shard-level baseline constraint (C7) and the normalized voting weight defined in Eq. (4) (with eligibility enforced in Eq. (7)).

Given the reputation ρ_n of individual peers within the network, the reputation of each shard k is derived as the average reputation of all peers assigned to that shard. This approach ensures that the overall reputation of a shard reflects the collective trustworthiness of its constituent peers. To achieve this, we calculate the shard reputation by averaging the reputations of all peers within the shard, weighted equally by the number of peers in the shard, denoted by $|k|$, which represents the cardinality of shard k, i.e., the number of peers within shard k. Similarly, by using Eq. (2), the reputation of shard k is calculated as:

$$\rho_k = \frac{1}{|k|} \sum_{n\in k} \rho_n, \forall k = 1, \ldots, K. \quad (3)$$

Therefore, the reputation of a shard is proportional to the reputation of the verifiers that validate the blocks formed within the shard. Note that blocks produced by a shard are added to their corresponding parallel chain only after all other nodes within that shard have validated them to ensure their validity before they can become invalid [59]. In particular, peers are required to agree on each block's authenticity inside their respective shards. As a starting point, the voting weight $w_n^{\text{vote}}(t)$ of peer N_n inside shard k is set proportional to its current reputation state, giving the unconstrained baseline form:

$$w_n^{\text{vote}}(t) = \frac{\rho_n(t)}{\sum_{m\in k} \rho_m(t)}, \ \forall n \in k, \quad (4)$$

such that peers with higher reputation in their shards have increased influence on block validity agreement. Consequently, any block produced by the participants of shard k is validated if $\sum_{n\in k} w_n^{\text{vote}}(t)\, a^n > 0.50$, where a^n denotes the agreement indicator.

The subjective-logic update in Eq. (1) is defined independently of the cloud, fog, and edge-tier classification. The architectural model in Section 3 already distinguishes these tiers in terms of authority and administrative control, with cloud nodes managed by the BaaS provider and edge nodes typically owned by third-party operators. The reputation system instead captures run-time behavioral evidence through positive and negative feedback and link quality, regardless of tier. As a result, nodes that consistently provide reliable service, including cloud controllers, retain high reputation scores. In contrast, nodes that misbehave, especially fog or edge peers that are more exposed to adversarial activity, accumulate disbelief and indecision. These nodes are gradually penalized as their reputation decays toward zero, suppressing their contribution to the shard reputation, and are excluded from consensus voting by the operative voting rule in Eq. (7).

We introduce a penalty mechanism for peers whose reputation falls below a threshold ρ^{th}. Specifically, after computing $\rho_n(t)$ at the end of timeslot t, we update the stored reputation used in the next timeslot as:

$$\rho_n(t{+}1) = \begin{cases} \max\{0, \rho_n(t) - \pi\}, & \rho_n(t) < \rho^{\text{th}}, \\ \rho_n(t), & \text{otherwise.} \end{cases} \tag{5}$$

where $\pi > 0$ is a penalty factor. The updated value $\rho_n(t{+}1)$ is used by the validator eligibility and voting rules in timeslot $t{+}1$. The threshold $\rho^{\text{th}} \in (0, 1)$ is a system-level parameter that separates persistently misbehaving peers from honest or occasionally uncertain ones. In practice, ρ^{th} is set to a moderate value between the steady-state reputations of honest and malicious nodes. In our experiments, we choose $\rho^{\text{th}} = 0.5$, consistent with the reputation threshold in Table 3. As shown in Section 6, honest peers converge to reputations close to 1, while malicious peers rapidly decay toward 0. Thus, the threshold retains a sufficient pool of honest validators in each shard and filters out only nodes that exhibit sustained misbehavior. With this update, peers with $\rho_n(t) < \rho^{\text{th}}$ are excluded from block validation, and voting weights are normalized over eligible validators. We distinguish between *shard membership* and *consensus participation*. The binary variable $\chi_{n,k} \in \{0, 1\}$ represents the shard membership of node n, where $\chi_{n,k} = 1$ indicates that node n belongs to shard k. Constraint (C8) in Section 4.4 enforces mutually exclusive shard memberships, that is, $\sum_{k\in\mathcal{K}} \chi_{n,k} = 1$, $\forall n \in \mathcal{N}$. The shard membership decision is specified in the joint resource and shard allocation problem formulated in Section 4.4.

To enforce security, the eligible validator set in shard k is defined based on the global reputation threshold ρ^{th} as:

$$\mathcal{N}_k^{\text{val}} \triangleq \left\{ n \in \mathcal{N} \mid \chi_{n,k} = 1,\ \rho_n(t) \geq \rho^{\text{th}} \right\},\ \forall k \in \mathcal{K}. \tag{6}$$

Nodes with $\rho_n(t) < \rho^{\text{th}}$ remain connected to the system, but they do not participate in block validation and have zero voting influence. We use $m \in k$ as shorthand for nodes assigned to shard k, i.e., $\{m \in \mathcal{N} \mid \chi_{m,k} = 1\}$. Thus, consensus participation is restricted to eligible validators in each shard, while shard-level BFT feasibility under the evaluated deployment is discussed in Section 6.

Since the unconstrained baseline in Eq. (4) does not exclude misbehaving nodes from voting, we now refine it by restricting the sum to the eligible validator set $\mathcal{N}_k^{\text{val}}$ defined in Eq. (6). Accordingly, the operative reputation-weighted voting rule inside shard k in timeslot t is defined as:

$$w_n^{\text{vote}}(t) = \frac{\rho_n(t)\, \mathbb{I}(\rho_n(t) \geq \rho^{\text{th}})}{\sum_{m\in k} \rho_m(t)\, \mathbb{I}(\rho_m(t) \geq \rho^{\text{th}})}, \quad \forall n \in k. \tag{7}$$

We assume each shard contains at least one eligible validator in the considered deployments, so the denominator remains non-zero. Now that we have already determined the reputation of a node ρ_n, the average reputation of a node in the system is calculated as:

$$\rho^{\text{avg}} = \frac{\sum_{n\in\mathcal{N}} \rho_n}{N} \tag{8}$$

The quantity ρ^{avg} is a system-wide statistic that summarizes the current trust level of all participating nodes. It serves as a baseline for monitoring and for reputation-aware recruitment rules. It does not specify shard indices. Shard membership is represented by $\chi_{n,k}$, which is defined in the problem formulation.

4.2. Communication Model

We consider the integration of edge, fog, and cloud computing capabilities to achieve low-latency computing, communication, and storage services. Network slicing is employed with the orchestration of SDN and NFV to provide services tailored to the specific requirements of different applications in accordance with ETSI [60]. Considering the dynamics of a MTC environment [3], we use $b_e(t)$ to indicate the network bandwidth during the timeslot t in the range of $[0, b_e^{\max}]$, where $b_e^{\max}$ is the maximum bandwidth capacity of link $e \in \mathcal{E}$. Services such as resource allocation, task offloading, and traffic management for different application classes are delivered through multiple virtualized network slices. Let l_j be the size of raw data required for provisioning service j where $j \in \mathcal{J}$. Here, $\mathcal{J}$ denotes the global catalog of service (or slice) types supported by the system. The same catalog is available in every shard. Depending on resource availability, each shard k can host instances of a subset of services $j \in \mathcal{J}$, so that the delay $D_{j,k}^{\text{total}}$ represents the performance of service j when it is provisioned by shard k. The data volume for service j is calculated as $l_j = \sum_{t=t_0}^{t'} r_j \Delta t$, where t_0 and t' are indicated as initial and completed timeslot and r_j is denoted as the data rate for service j. Hence, the transmission delay is calculated as $(t' - t_0)\, \Delta t = l_j / r_j$.

In an MTC environment, a higher data rate reduces the raw data retrieval time since the transmission delay satisfies l_j / r_j. However, the aggregate data rate over each link is constrained by the available bandwidth, and violating this capacity leads to backlog and delay inflation. Hence, we impose Eq. (9) to

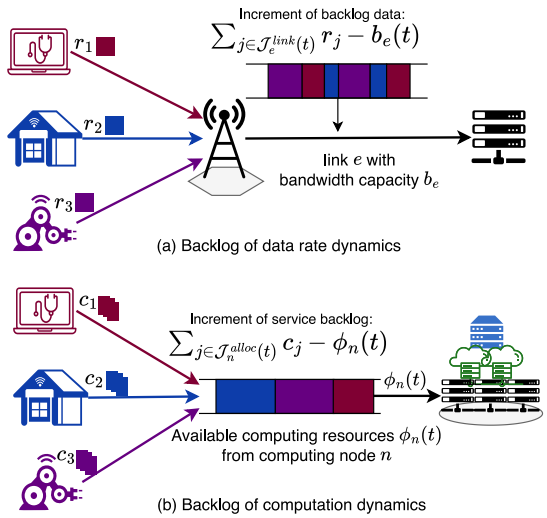


Figure 4: Queueing solution for data forwarding and computation resources.

enforce the link-capacity limit during timeslot t while enabling higher feasible per-service data rates.

$$\sum_{j\in\mathcal{J}_e^{\text{link}}(t)} r_j \le b_e(t),\ \forall e \in \mathcal{E}, \tag{9}$$

where $\mathcal{J}_e^{\text{link}}(t)$ is the set of provisioned services that utilize communication link e. Fig. 4a shows the virtual queues of data rate, where $q_e(t)$ is the length of virtual queue that represents the backlog of data that is waiting to be forwarded over communication link e at timeslot t. Here, the arrival and departure rate of queue is represented as the overall data rate of service j utilizing communication link e and the available network bandwidth $b_e(t)$ at timeslot t, respectively. Hence, the dynamics of the virtual queue over a certain period of $q_e(t)$ at slot t is presented as:

$$q_e(t+1) = q_e(r_j) = \left[q_e(t) + \sum_{j\in\mathcal{J}_e^{\text{link}}(t)} r_j - b_e(t) \right]^+,\ \forall e \in \mathcal{E}, \tag{10}$$

where $[h]^+ = \max[0, h]$. To ensure that all data for service j is forwarded within timeslot t through communication link e, we require the following constraint to show that $q_e(t)$ is stable:

$$\sum_{j\in\mathcal{J}_e^{\text{link}}(t)} r_j \le \frac{1}{T}\sum_{t=1}^{T} b_e(t),\ e \in \mathcal{E}. \tag{11}$$

4.3. Computation Model

Recall that $N_{\mathbb{E}}, N_{\mathbb{F}}, N_{\mathbb{C}}$ are the computing nodes (CNs), acting as the validators which validate other peers depending on their reputation in the system. We use a time-dependent variable $\phi_n(t)$ to denote the available computing capacity (measured by the maximum number of CPU cycles that CN n can execute in one timeslot) of the peers. Due to the service commercialization, the availability of computing resources for fog and cloud servers is more stable as compared to edge servers in terms of delay-sensitive and resource-constraint services [61]. Due to the limited resource and CPU availability, the computation capacity for edge servers vary with the time as they only utilize the unused resources. Therefore, the computing resources of edge servers $\phi_n(t), n \in N_{\mathbb{E}}$ is the random variable within timeslot t. Without loss of generality, $\phi_o(t) \gg \phi_m(t) \gg \phi_n(t)$ where $o \in N_{\mathbb{C}}, m \in N_{\mathbb{F}}, n \in N_{\mathbb{E}}$, since BaaS servers are resource rich and have more computational capacity as compared to fog and edge servers. The CN starts to provide service j after receiving the slice request from a random peer, acting as tenant. Each slicing request for a service j is characterized by the tuple $\{c_j, l_j\}$, where c_j is denoted as the number of CPU cycles required to provide service j. Hence, the processing delay of service j allocated to CN n is represented as the sum of forwarding latency and completion time. Meanwhile, the input queue is the total computing resource required for incoming service at timeslot t and the queue departure rate is equal to $\phi_n(t)$ (as shown in Fig. 4b). Let $Q_n(t)$ be the backlog of queued service at CN n and the dynamics of $Q_n(t)$ is represented as:

$$Q_n(t+1) = Q_n(c_j) = \left[Q_n(t) + \sum_{j\in\mathcal{J}_n^{\text{alloc}}(t)} c_j - \phi_n(t) \right]^+,\ \forall n \in \mathcal{N}, \tag{12}$$

where $[x]^+ = \max[0, x]$ and $\mathcal{J}_x^{\text{alloc}}(t)$ is the set of services allocated to CNs. Here, $Q_n(t)$ and c_j are measured in CPU cycles, while $\phi_n(t)$ denotes the maximum number of CPU cycles that CN n can process within one timeslot, so the subtraction in Eq. (12) represents the amount of queued work that can be served during that interval. Based on this workload queue, the total processing delay of a service in shard k is calculated as:

$$D_{j,k}^{\text{proc}}(c_j) = \frac{Q_n(t) + c_j}{\phi_n(t)}. \tag{13}$$

By combining the processing delay and data rate, we represent the service completion delay (i.e., time to complete the service when it is executed and return its result) for each service j in shard k as:

$$D_{j,k}^{\text{comp}}(r_j) = D_{j,k}^{\text{proc}}(c_j) + \frac{l_j}{\sum_{j\in\mathcal{J}_e^{\text{link}}(t)} r_j}. \tag{14}$$

4.4. Problem Formulation

To calculate the block verification delay of a shard denoted as $D_{j,k}^{\text{block}}$, we consider the number of service instances, the computational power of the verifying nodes (since verification involves computational tasks such as verification of the data integrity and the validity of the signatures), the size of the block to be verified (larger blocks would take longer to verify), and the network's overall data rate. Additionally, the block verification delay is influenced by the data rate r_j, which defines how quickly data can be transmitted across the network. A higher rate typically contributes to a shorter block verification delay. Taking all these factors into account, we formulate the block verification delay as follows:

$$D_{j,k}^{\text{block}}(c_j) = \frac{l_{j,k}^{\text{block}}}{\sum_{j\in\mathcal{J}_e^{\text{link}}(t)} r_j} + \frac{c_j}{\phi_n(t)}, \tag{15}$$

where $l_{j,k}^{\text{block}}$ represents the total size of the block data to be verified, which increases with the number of service instances. This increase in size directly impacts the time required for block verification. Now we represent the total delay of service j in shard k as:

$$D_{j,k}^{\text{total}}(r_j, c_j) = D_{j,k}^{\text{proc}}(c_j) + D_{j,k}^{\text{comp}}(r_j) + D_{j,k}^{\text{block}}(c_j) \tag{16}$$

Recall that CNs are required to pay a certain amount in fees to the BaaS provider for necessary computation tasks, service provisioning, authentication, and verification process. Hence, we incentivize more CNs to participate in the proposed system as they can process more tasks without having to pay the additional fees by minimizing the processing fee, completion delay and computational resource usage for CNs.

The service provision cost for service j is represented as $\zeta_j c_j$, where ζ_j is the price that is charged for computational resource usage for service j. We define the total reputation of shard k as $\rho_k^{\text{total}} = \sum_{n\in\mathcal{N}} \rho_n \chi_{n,k}$, which aggregates the reputations of all computing nodes assigned to shard k. This definition is consistent with the average shard reputation in Eq. (3), since $\rho_k^{\text{total}} = |k|\rho_k$ with $|k|$ denoting the number of peers in shard k. By combining the aggregate delay term $\sum_{j\in\mathcal{J}} \sum_{k\in\mathcal{K}} D_{j,k}^{\text{total}}$, the total shard reputation term $\sum_{k\in\mathcal{K}} \rho_k^{\text{total}}$, and the resource provision cost term $\sum_{j\in\mathcal{J}} \zeta_j c_j$, we obtain the objective function $\mathcal{F}_0$ as:

$$\mathcal{F}_0(r, c, \chi) = w_1 \sum_{j\in\mathcal{J}} \sum_{k\in\mathcal{K}} D_{j,k}^{\text{total}} - w_2 \sum_{k\in\mathcal{K}} \rho_k^{\text{total}} + w_3 \sum_{j\in\mathcal{J}} \zeta_j c_j. \tag{17}$$

Here, $j \in \mathcal{J}$ indexes end-to-end services (or slices) such as e-health, smart home, and IIoT that the system supports at the global level, while $k \in \mathcal{K}$ indexes blockchain shards. Each shard provisions a subset of services from $\mathcal{J}$ according to its resource state and slice requirements. The term $D_{j,k}^{\text{total}}$ captures the contribution of shard k to the end-to-end delay of service j when shard k participates in its provisioning. In contrast, ρ_k^{total} aggregates the reputation of all validators in shard k and does not depend on the specific service being processed. The cost term $\zeta_j c_j$ only depends on the computing resources allocated to service j. Separating the sums in Eq.(17) prevents artificial amplification of shard reputation or computing cost when the system supports multiple services. Note that, w_1, w_2, w_3 are different weights such that $w_1 + w_2 + w_3 = 1$. These weights represent the service provider's relative emphasis on (i) low latency ($D_{j,k}^{\text{total}}$), (ii) high system-wide reputation (ρ_k^{total}), and (iii) low operational cost ($\zeta_j c_j$), respectively. The weight vector (w_1, w_2, w_3) is selected *offline* during slice instantiation, guided by slice-specific SLAs. For instance, latency-critical verticals such as telesurgery typically assign greater emphasis to w_1; safety-critical verticals (e.g., connected-vehicle platooning) prioritize a larger w_2 to favor highly trusted nodes; and cost-sensitive, best-effort telemetry applications increase w_3. For all experiments presented in Section 6, we adopt a balanced configuration $(w_1, w_2, w_3) = (0.4, 0.4, 0.2)$, reflecting the latency versus security considerations highlighted by recent ETSI guidelines on 5G/6G service orchestration. Operators can adjust these coefficients through an offline grid search over the two-simplex subject to KPI thresholds, without altering any other component of the optimization framework. Thus, the proposed solution remains applicable for any combination of non-negative weights summing to one.

The system is designed with an optimization goal to achieve balanced load distribution across heterogeneous shards within each tier, where each shard k comprises CNs with varying capacities ϕ_n as specified in constraint (C3). To achieve this load-balanced state while respecting per-node capacity limits, we introduce the forwarding rate vector $\mathbf{r} = (r_1, r_2, \ldots, r_j)$ and the required computing resource vector $\mathbf{c} = (c_1, c_2, \ldots, c_j)$. We focus on maximizing the minimum objective to ensure proportional workload distribution. Therefore, the final formulation of our reputation-enabled shard allocation problem is expressed as:

$$\begin{aligned}
\mathbf{P1}: \quad & \max_{\mathbf{r},\mathbf{c},\chi} \min \mathcal{F}_0(r, c, \chi) \\
& = w_1 \sum_{j\in\mathcal{J}} \sum_{k\in\mathcal{K}} D_{j,k}^{\text{total}} - w_2 \sum_{k\in\mathcal{K}} \rho_k^{\text{total}} + w_3 \sum_{j\in\mathcal{J}} \zeta_j c_j \\
\text{s.t.} \quad & \text{(C1)}: \sum_{j\in\mathcal{J}_e^{\text{link}}(t)} r_j \le b_e(t), \forall e \in \mathcal{E}, \forall t \in 0, \ldots, T, \\
& \text{(C2)}: \sum_{j\in\mathcal{J}_e^{\text{link}}(t)} r_j \le \tfrac{1}{T} \sum_{t=1}^{T} b_e(t), \forall e \in \mathcal{E}, \\
& \text{(C3)}: \sum_{j\in\mathcal{J}} c_j \le \phi_n, \forall n \in \mathcal{N}, \\
& \text{(C4)}: \sum_{j\in\mathcal{J}_e^{\text{link}}(t)} r_j \le c_j, \forall n \in \mathcal{N}, \\
& \text{(C5)}: r_j \ge 0, c_j \ge 0, \forall j \in \mathcal{J}, \\
& \text{(C6)}: \chi_{n,k} \in \{0, 1\}, \forall n, k, \\
& \text{(C7)}: \textstyle\sum_{n\in\mathcal{N}} \chi_{n,k} (\rho_n - \rho^{\text{avg}}) \ge 0, \quad \forall k \in \mathcal{K}, \\
& \text{(C8)}: \textstyle\sum_{k\in\mathcal{K}} \chi_{n,k} = 1, \forall n \in \mathcal{N}
\end{aligned} \tag{18}$$

where, (C1) ensures that the data rate for each tenant does not exceed the available bandwidth, while (C2) ensures the total amount of data forwarded by the tenants connected to each link remains within the average bandwidth available over the given timeslot. Besides, (C3) specifies that the computation assigned to each CN does not exceed its available resources. Consequently, (C4) guarantees that the data received by the CNs from the tenants does not exceed their computing resources. This constraint ensures that the CNs cache the data received from the tenants and serve subsequent requests from their cache, which can significantly reduce the data transmission and processing costs. Furthermore, (C5) represents the non-negative parameters for data rate and required computing resources, and (C6) defines the binary decision variable for shard assignment, where $\chi_{n,k} = 1$ indicates that node n participates in shard k and $\chi_{n,k} = 0$ otherwise. (C7) ensures that the average reputation of nodes assigned to each shard remains no smaller than the system-wide baseline ρ^{avg}, and (C8) partitions the validator set into K disjoint committees by assigning every node to exactly one shard. The minimum number of peers per shard is ensured by the shard-formation procedure in the system model, rather than imposed directly in P1. The formulated problem P1 is a *non-convex mixed-integer nonlinear programming (MINLP) problem* due to two aspects. Firstly, the objective function $\mathcal{F}_0$ contains a max-min operation that fundamentally lacks convexity. Secondly, the binary constraint imposed on the decision variable $\chi_{n,k}$ in (C6) designates the problem as mixed integer. Although (C1) to (C5)

and (C8) outline a convex feasible region, the introduction of mixed-integer (C6) and nonlinear (C7) components contributes to the problem's non-convexity.

5. ADMM-based Joint Resource and Shard Allocation Solution

In this section, we present a distributed solution to the optimization problem formulated in the previous section. We first transform the intractable MINLP model into a convex form through relaxation and epigraph techniques. Subsequently, we propose a reputation-aware ADMM algorithm to jointly allocate resources and assign shards efficiently.

5.1. Problem Transformation and Convex Relaxation

To render the optimization problem tractable **P1**, we first address the non-convexity arising from the max-min objective and the fractional delay terms. We first introduce a new variable, γ, to transform the max-min problem into a more tractable maximization problem using the epigraph form. The epigraph transformation is used to convert the max-min problem into a simpler minimization problem by introducing an auxiliary variable that bounds the objective function from below. This simplification allows for easier application of standard optimization techniques. By applying $\mathcal{F}_1 = -\gamma$, we convert the maximization problem into a standard minimization problem, denoted as **P2**, as follows:

$$\begin{aligned} \mathbf{P2}: \quad & \min_{\mathbf{r},\mathbf{c},\boldsymbol{\chi},\gamma,\mathbf{u},\mathbf{v}} \; -\gamma \\ \text{s.t.} \quad & \text{(C1)-(C8)}, \;\; \text{(C9)}: \; \gamma \le \tilde{\mathcal{F}}_0(\mathbf{u},\mathbf{v},\mathbf{c},\boldsymbol{\chi}). \end{aligned} \tag{19}$$

This transformation is crucial as it redefines the problem into a form that is more conducive to continuous optimization methods. However, for algorithmic tractability, the feasible set must be convex. The delay terms in $\mathcal{F}_0$ contain reciprocal expressions such as l_j/r_j and $L_{e,t}/s_{e,t}$, which create nonconvex hypographs in constraint (C9). Here, $L_{e,t}$ denotes the aggregate data volume (in bits) that must be transmitted over communication link e during timeslot t. It aggregates the data sizes l_j of all services that use link e in that slot. The term $s_{e,t}$ represents the corresponding aggregate data rate on link e and is defined as $s_{e,t} = \sum_{j \in \mathcal{J}_e^{\text{link}}(t)} r_j$, consistent with the communication model in Section 4.2. To address this issue, we introduce additional epigraph variables to linearize $\mathcal{F}_0$ while preserving convexity:

$$\begin{aligned} &\text{(C10)}: \quad 2r_j u_j \;\ge\; l_j^2, \quad r_j \ge 0, \; u_j \ge 0, \;\; \forall j, \\ &\text{(C11)}: \quad 2s_{e,t} v_{e,t} \;\ge\; L_{e,t}^2, \quad s_{e,t} = \sum_{j \in \mathcal{J}_e^{\text{link}}(t)} r_j, \\ &\qquad\qquad s_{e,t} \ge 0, \; v_{e,t} \ge 0, \;\; \forall e,t. \end{aligned}$$

Constraints (C10)-(C11) are rotated second-order cone (RSOC) epigraphs that implement the relationships $u_j \ge l_j^2/(2r_j)$ and $v_{e,t} \ge L_{e,t}^2/(2s_{e,t})$ in a convex manner. With these auxiliary variables, the performance functional becomes affine:

$$\begin{aligned} \tilde{\mathcal{F}}_0(\mathbf{u},\mathbf{v},\mathbf{c},\boldsymbol{\chi}) \;=\; & \sum_j \alpha_j u_j \;+\; \sum_{e,t} \kappa_{e,t} v_{e,t} \\ & + \sum_j \omega_j^{(c)} c_j \;-\; \sum_{n,k} \omega^{(\rho)} \rho_n \chi_{n,k}, \end{aligned} \tag{20}$$

where the RSOC weights are calibrated for exact equivalence to the original objective:

$$\alpha_j = \frac{2\omega_j^{(r)}}{l_j}, \qquad \kappa_{e,t} = \frac{2\omega_{e,t}^{(\ell)}}{L_{e,t}}, \tag{21}$$

ensuring that minimizing $-\gamma$ with $\gamma \le \tilde{\mathcal{F}}_0$ becomes exactly equivalent to the original max-min formulation. By replacing $\mathcal{F}_0$ with $\tilde{\mathcal{F}}_0$ in (C9), the hypograph constraint $\gamma \le \tilde{\mathcal{F}}_0(\cdot)$ becomes convex due to the affine right-hand side.

We decompose the minimization **P2** into a primal problem **P3** by relaxing the integer constraints to make all variables in **P2** continuous. Relaxing the integer constraints allows the problem to be addressed using continuous optimization techniques, which are often more tractable. While this relaxation may introduce some approximation, it often provides a solution close to the optimal integer values, which can be refined later if necessary. To preserve the exclusivity structure for ADMM while removing integrality, we relax $\chi_{n,k} \in \{0,1\}$ to a row-simplex. The relaxed problem is presented as:

$$\begin{aligned} \mathbf{P3}: \quad & \min\nolimits_{\mathbf{r},\mathbf{c},\boldsymbol{\chi},\gamma,\mathbf{u},\mathbf{v}} \mathcal{F}_1(r,c,\chi) \\ \text{s.t.} \quad & \text{(C1) to (C9) with } \tilde{\mathcal{F}}_0, \;\; \text{(C10)-(C11)}, \\ & \text{(C12)}: \; 0 \le \chi_{n,k} \le 1, \;\; \textstyle\sum_k \chi_{n,k} = 1, \quad \forall n. \end{aligned} \tag{22}$$

Constraint (C12) is the convex relaxation of the integer shard-assignment constraints (C6) and (C8) in **P1**: in the original MINLP each node participates in exactly one shard through a binary one-hot vector $(\chi_{n,1},\ldots,\chi_{n,K})$, whereas in **P3** we allow fractional values during optimization and subsequently round the solution to recover integral shard memberships.

We refine the compute constraint (C3) to correctly model distributed service placement. The original constraint $\sum_j c_j \le \phi_n$ unrealistically assumes every node must handle all services. We replace it with:

$$\text{(C3}'\text{)}: \quad \sum_j c_{j,n} \le \phi_n, \;\; \forall n; \qquad \sum_n c_{j,n} = c_j, \;\; c_{j,n} \ge 0, \tag{23}$$

where $c_{j,n}$ represents the compute resources allocated to service j on node n. This allows flexible distributed service placement across the multi-tier infrastructure. The relaxation of **P3** into a continuous problem with convex constraints facilitates the use of efficient optimization algorithms that can handle large-scale problems while providing convergence guarantees.

5.2. Lagrangian Dual Decomposition

Building on the continuous relaxation in **P3**, we adopt a Lagrangian dual optimization approach to further decompose the problem and enable a distributed solution. The constraints

(C1)-(C2), (C3′), (C5), and (C7)-(C12) are rewritten in the standard $g(x) \leq 0$ form for dual decomposition, where (C9) captures the affine epigraph of $\widetilde{\mathcal{F}}_0$, (C10)-(C11) model the RSOC epigraphs, and (C12) encodes the relaxed shard-assignment simplex structure.

We introduce the Lagrange multipliers $\xi_{e,t}$, Ξ_e, ν_x, $\varpi_{e,t}$, Ψ_j, $\eta_{n,k}$, and θ to incorporate the respective constraints into the objective function and to facilitate the application of Lagrangian dual decomposition. For clarity, we represent these multipliers collectively as vectors in the Lagrangian function: $\boldsymbol{\xi} = \{\xi_{e,t} \mid \forall e,t\}$, $\boldsymbol{\Xi} = \{\Xi_e \mid \forall e\}$, $\boldsymbol{\nu} = \{\nu_x \mid \forall x\}$, $\boldsymbol{\varpi} = \{\varpi_{e,t} \mid \forall e,t\}$, $\boldsymbol{\Psi} = \{\Psi_j \mid \forall j\}$, $\boldsymbol{\eta} = \{\eta_{n,k} \mid \forall n,k\}$, and $\boldsymbol{\theta} = \{\theta\}$.

The Lagrangian function incorporating all constraint multipliers is:

$$\mathcal{L}(\mathbf{r},\mathbf{c},\boldsymbol{\chi},\gamma,\mathbf{u},\mathbf{v},\boldsymbol{\lambda}) = \mathcal{F}_1(\mathbf{r},\mathbf{c},\boldsymbol{\chi}) + \boldsymbol{\lambda}^T\mathbf{g}(\mathbf{r},\mathbf{c},\boldsymbol{\chi},\gamma,\mathbf{u},\mathbf{v}), \quad (24)$$

where $\boldsymbol{\lambda} = (\boldsymbol{\xi},\boldsymbol{\Xi},\boldsymbol{\nu},\boldsymbol{\varpi},\boldsymbol{\Psi},\boldsymbol{\eta},\boldsymbol{\theta})$ are the multipliers and $\mathbf{g}(\cdot)$ represents the constraint vector. To formulate the dual problem for the optimization problem **P3**, we derive the dual function by taking the infimum of the Lagrangian function over the primal variables. The infimum represents the best (i.e., lowest) value that the Lagrangian function achieves given the dual variables to ensure that the primal constraints are respected while simplifying the problem. The dual function is defined as:

$$\begin{aligned} &g(\boldsymbol{\xi},\boldsymbol{\Xi},\boldsymbol{\nu},\boldsymbol{\varpi},\boldsymbol{\Psi},\boldsymbol{\eta},\boldsymbol{\theta}) \\ &\quad = \inf_{\mathbf{r},\mathbf{c},\boldsymbol{\chi},\gamma,\mathbf{u},\mathbf{v}} \mathcal{L}(\mathbf{r},\mathbf{c},\boldsymbol{\chi},\gamma,\mathbf{u},\mathbf{v},\boldsymbol{\xi},\boldsymbol{\Xi},\boldsymbol{\nu},\boldsymbol{\varpi},\boldsymbol{\Psi},\boldsymbol{\eta},\boldsymbol{\theta}). \end{aligned} \quad (25)$$

The dual problem aims to maximize this dual function subject to constraints that the dual variables are non-negative (for inequality constraints in the primal problem). The dual problem is presented as:

$$\begin{aligned} \mathbf{P4}: \quad &\max_{\boldsymbol{\xi},\boldsymbol{\Xi},\boldsymbol{\nu},\boldsymbol{\varpi},\boldsymbol{\Psi},\boldsymbol{\eta},\boldsymbol{\theta}} \; g(\boldsymbol{\xi},\boldsymbol{\Xi},\boldsymbol{\nu},\boldsymbol{\varpi},\boldsymbol{\Psi},\boldsymbol{\eta},\boldsymbol{\theta}) \\ \text{s.t.} \quad &\xi_{e,t},\Xi_e,\nu_x,\varpi_{e,t},\Psi_j,\eta_{n,k},\theta \geq 0. \end{aligned} \quad (26)$$

We apply the iterative ascent method to generate the dual variables. We apply iterative ascent to solve the dual problem with step size $\sigma(i)$:

$$\boldsymbol{\lambda}^{(i+1)} = \left[\boldsymbol{\lambda}^{(i)} + \sigma(i)\nabla_{\boldsymbol{\lambda}} g(\boldsymbol{\lambda}^{(i)})\right]^+, \quad (27)$$

where $[h]^+ \triangleq \max\{0,h\}$, (i) denotes the number of iterations, $\sigma(i)$ is the step size, and $\frac{d}{di}$ represents the derivative with respect to the iteration number i. The step size $\sigma(i)$ plays a critical role in ensuring the convergence of the iterative method. It is typically chosen to decrease over time and allow for gradual refinement of the dual variables.

5.3. Reputation-Aware ADMM with Consensus Splitting

To operationalize the relaxed problem **P3** in a scalable manner and exploit the multi-tier structure, we employ the Alternating Direction Method of Multipliers (ADMM) with consensus splitting. We reformulate **P3** by introducing auxiliary variables to enable decomposition. We define the stacked primal vector $\mathbf{x} = (\mathbf{r},\mathbf{c},\mathbf{C},\boldsymbol{\chi},\gamma,\mathbf{u},\mathbf{v})$ and an auxiliary copy $\mathbf{z}$ of the same dimension, where $\mathbf{C} = \{c_{j,n}\}$ denotes the matrix of distributed compute allocations. Let $f(\mathbf{x}) = \iota_C(\mathbf{x})$ denote the indicator function where C collects all convex constraints of **P3**, i.e., (C1)-(C2), (C3′), (C4)-(C5), and (C7)-(C12), and set $g(\mathbf{z}) = \mathcal{F}_1(\mathbf{z}) = -z_\gamma$. The consensus form becomes:

$$\mathbf{P5}: \quad \min_{\mathbf{x},\mathbf{z}} f(\mathbf{x}) + g(\mathbf{z}) \quad \text{subject to} \quad \mathbf{x} = \mathbf{z}. \quad (28)$$

Using the scaled dual variable $\mathbf{y} = \boldsymbol{\lambda}/\varrho$ with ADMM penalty parameter $\varrho > 0$ (to avoid collision with reputation symbols ρ_n), the scaled augmented Lagrangian is:

$$\mathcal{L}_\varrho(\mathbf{x},\mathbf{z},\mathbf{y}) = f(\mathbf{x}) + g(\mathbf{z}) + \frac{\varrho}{2}\|\mathbf{x}-\mathbf{z}+\mathbf{y}\|_2^2 - \frac{\varrho}{2}\|\mathbf{y}\|_2^2. \quad (29)$$

At each iteration ℓ, the ADMM updates are performed as follows. In the primal variable update (x-update), we solve the projection step:

$$\mathbf{x}^{(\ell+1)} = \arg\min_{\mathbf{x}\in C} \frac{\varrho}{2}\|\mathbf{x} - (\mathbf{z}^{(\ell)} - \mathbf{y}^{(\ell)})\|_2^2. \quad (30)$$

This is the Euclidean projection of $(\mathbf{z}^{(\ell)} - \mathbf{y}^{(\ell)})$ onto the constraint set C. The projection decomposes into separable blocks due to the constraint structure, enabling efficient parallel computation. For resource allocation variables $(\mathbf{r},\mathbf{c})$, we solve:

$$(\mathbf{r}^{(\ell+1)},\mathbf{c}^{(\ell+1)}) = \Pi_{\mathcal{RC}}\left[(\mathbf{z}_r^{(\ell)},\mathbf{z}_c^{(\ell)}) - (\mathbf{y}_r^{(\ell)},\mathbf{y}_c^{(\ell)})\right], \quad (31)$$

where $\Pi_{\mathcal{RC}}$ denotes projection onto the set defined by constraints (C1), (C2), (C3′), (C4), and (C5). For shard assignments $\boldsymbol{\chi}$, we perform row-wise simplex projection:

$$\chi_{n,:}^{(\ell+1)} = \Pi_{\Delta_K}\left[\mathbf{z}_{\chi,n,:}^{(\ell)} - \mathbf{y}_{\chi,n,:}^{(\ell)}\right], \quad (32)$$

where Π_{Δ_K} projects onto the probability simplex intersected with $[0,1]^K$ in accordance with (C12). For the epigraph variable γ, projection onto constraint (C9) with $\widetilde{\mathcal{F}}_0$ gives:

$$\gamma^{(\ell+1)} = \min\{z_\gamma^{(\ell)} - y_\gamma^{(\ell)}, \widetilde{\mathcal{F}}_0(\mathbf{u}^{(\ell+1)},\mathbf{v}^{(\ell+1)},\mathbf{c}^{(\ell+1)},\boldsymbol{\chi}^{(\ell+1)})\}. \quad (33)$$

In the auxiliary variable update (z-update), we apply the proximal operator:

$$\mathbf{z}^{(\ell+1)} = \text{prox}_{\frac{1}{\varrho}g}(\mathbf{x}^{(\ell+1)} + \mathbf{y}^{(\ell)}). \quad (34)$$

Since $g(\mathbf{z}) = -z_\gamma$ is linear in z_γ, its proximal operator is closed-form:

$$\begin{aligned} &\mathbf{z}^{(\ell+1)} = \mathbf{x}^{(\ell+1)} + \mathbf{y}^{(\ell)} + \frac{1}{\varrho}\mathbf{e}_\gamma, \\ \text{i.e.,} \quad &z_\gamma^{(\ell+1)} = x_\gamma^{(\ell+1)} + y_\gamma^{(\ell)} + \frac{1}{\varrho}, \quad z_q^{(\ell+1)} = x_q^{(\ell+1)} + y_q^{(\ell)} \; \forall q \neq \gamma. \end{aligned} \quad (35)$$

Finally, the dual variable update follows:

$$\mathbf{y}^{(\ell+1)} = \mathbf{y}^{(\ell)} + (\mathbf{x}^{(\ell+1)} - \mathbf{z}^{(\ell+1)}). \quad (36)$$

To obtain feasible binary assignments from the relaxed solution, we formulate a reputation-guided minimum-cost

assignment problem. We define the cost matrix incorporating reputation bias as

$$C_{n,k} = -\omega^{(\rho)}\rho_n, \tag{37}$$

where the cost coefficient reflects the reputation impact on the χ-dependent part of $\tilde{\mathcal{F}}_0$. Since $\mathcal{F}_1 = -\gamma$ does not depend directly on χ, the assignment cost comes from the χ-dependent part of $\tilde{\mathcal{F}}_0$. We solve the integer program:

$$\hat{\boldsymbol{\chi}} = \arg\min_{\chi\in\{0,1\}^{N\times K}} \sum_{n,k} C_{n,k}\chi_{n,k} \quad \text{s.t.} \quad \sum_k \chi_{n,k} = 1. \tag{38}$$

This minimum-cost flow problem enforces the integrality and one-shard-per-node structure of (C6) and (C8), while respecting shard capacity limits C_k that are consistent with the shard-formation procedure in the system model. After obtaining $\hat{\boldsymbol{\chi}}$, we resolve for optimal resources:

$$(\hat{\mathbf{r}}, \hat{\mathbf{c}}, \hat{\gamma}) = \arg\min_{(\mathbf{r},\mathbf{c},\gamma)\in\mathcal{C}} \mathcal{F}_1(\mathbf{r}, \mathbf{c}, \hat{\boldsymbol{\chi}}, \gamma). \tag{39}$$

5.4. Convergence and Complexity Analysis

Here, we formally characterize the theoretical performance of the proposed algorithm. We establish the convergence properties of the ADMM iterative process and provide a binary recovery mechanism. Finally, we analyze the computational complexity to demonstrate the algorithm's scalability for real-time operations in large-scale multi-tier networks.

We now establish the theoretical convergence guarantees and complexity analysis for our ADMM-based approach. Algorithm 1 details our reputation-aware ADMM solution through a streamlined three-stage iterative process. The *Block-Structured Projection Stage* (line 4) computes decomposed Euclidean projections onto the constraint set C, including rates/links, RSOC epigraphs, compute variables, assignment simplexes, and epigraph constraints. The *Closed-Form Proximal Update Stage* (line 5) applies the proximal operator for the linear objective $g(\mathbf{z}) = -z_\gamma$. The *Dual Update and Adaptation Stage* (lines 6-8) adjusts dual variables and adaptively balances primal-dual residuals through penalty parameter tuning.

The reputation-guided binary recovery mechanism (lines 11-12) converts the relaxed solution to feasible integer assignments that satisfy the shard-assignment constraints (C6) and (C8) through minimum-cost flow optimization. The algorithm computes cost coefficients directly from normalized reputation scores, so highly reputed nodes are preferred during the discrete assignment process, maintaining the security properties of the multi-tier system while achieving near-optimal resource allocation.

Proposition 1. *The reputation-aware ADMM algorithm provides convergent solutions to the relaxed optimization problem* **P3** *while the binary recovery mechanism ensures feasible integer assignments for shard allocation under reputation constraints.*

Proof. The convergence follows from standard two-block ADMM theory with consensus splitting. Since the indicator function $f(\mathbf{x}) = \iota_C(\mathbf{x})$ is proper, closed, and convex, and $g(\mathbf{z}) = -z_\gamma$ is proper, closed, and convex, a feasible solution exists, and

Algorithm 1: Reputation-Aware ADMM for Shard Allocation

Input: Network $\mathcal{G}$, services $\mathcal{J}$, reputation scores $\{\rho_n\}$, penalty $\varrho > 0$, tolerance ϵ
Output: Resource allocation $(\hat{\mathbf{r}}, \hat{\mathbf{c}})$ and shard assignments $\hat{\boldsymbol{\chi}}$

`// Initialize variables`
1 Initialize $\mathbf{x}^{(0)}, \mathbf{z}^{(0)}, \mathbf{y}^{(0)}$
2 Set $\ell = 0$
`// Main ADMM iterations`
3 **repeat**
  `// Step 1: Block-structured projection onto constraint set C`
4 $\mathbf{x}^{(\ell+1)} = \arg\min_{\mathbf{x}\in C} \frac{\varrho}{2}\|\mathbf{x} - (\mathbf{z}^{(\ell)} - \mathbf{y}^{(\ell)})\|_2^2$
  `// Includes: rates/links, RSOC epigraphs, compute, assignments, epigraph`
  `// Step 2: Closed-form proximal operator update`
5 $\mathbf{z}^{(\ell+1)} = \mathbf{x}^{(\ell+1)} + \mathbf{y}^{(\ell)} + \frac{1}{\varrho}\mathbf{e}_\gamma$
  `// Step 3: Dual variable update`
6 $\mathbf{y}^{(\ell+1)} = \mathbf{y}^{(\ell)} + (\mathbf{x}^{(\ell+1)} - \mathbf{z}^{(\ell+1)})$
  `// Compute residuals and adapt penalty parameter`
7 $\mathbf{r}_{\text{pri}}^{(\ell+1)} = \mathbf{x}^{(\ell+1)} - \mathbf{z}^{(\ell+1)}$, $\mathbf{r}_{\text{dual}}^{(\ell+1)} = \varrho(\mathbf{z}^{(\ell+1)} - \mathbf{z}^{(\ell)})$
8 Update penalty ϱ based on residual balance
9 Set $\ell = \ell + 1$
10 **until** $\|\mathbf{r}_{pri}^{(\ell+1)}\|_2 \le \varepsilon_{pri}$ *and* $\|\mathbf{r}_{dual}^{(\ell+1)}\|_2 \le \varepsilon_{dual}$
`// Binary recovery via reputation-guided minimum-cost flow`
11 $\hat{\boldsymbol{\chi}} = \arg\min_{\chi\in\{0,1\}^{N\times K}} \sum_{n,k}(-\omega^{(\rho)}\rho_n)\chi_{n,k}$
12 $(\hat{\mathbf{r}}, \hat{\mathbf{c}}, \hat{\gamma}) = \arg\min_{(\mathbf{r},\mathbf{c},\gamma)\in C} -\gamma$
13 **return** $(\hat{\mathbf{r}}, \hat{\mathbf{c}}, \hat{\boldsymbol{\chi}})$

the x- and z-updates are solved exactly, the ADMM iterates satisfy $\|\mathbf{r}_{\text{pri}}^{(\ell)}\|_2 \to 0$ and $\|\mathbf{r}_{\text{dual}}^{(\ell)}\|_2 \to 0$, where $\mathbf{r}_{\text{pri}}^{(\ell)} = \mathbf{x}^{(\ell)} - \mathbf{z}^{(\ell)}$ and $\mathbf{r}_{\text{dual}}^{(\ell)} = \varrho(\mathbf{z}^{(\ell)} - \mathbf{z}^{(\ell-1)})$. Any accumulation point is primal-dual optimal for **P3**.

For binary recovery, the minimum-cost flow formulation with reputation bias ensures feasible integer assignments. To clarify the bound $\|\hat{\boldsymbol{\chi}} - \boldsymbol{\chi}^*\|_1 \le 2N$, note that for each node n, the relaxed solution vector $\boldsymbol{\chi}_n^*$ satisfies the simplex constraint of (C12), i.e., $0 \le \chi_{n,k}^* \le 1$ and $\sum_k \chi_{n,k}^* = 1$, which implies $\|\boldsymbol{\chi}_n^*\|_1 = 1$. Similarly, the recovered binary solution $\hat{\boldsymbol{\chi}}_n$ assigns node n to exactly one shard, so $\|\hat{\boldsymbol{\chi}}_n\|_1 = 1$. By the ℓ_1 triangle inequality, the deviation for a single node is bounded as

$$\|\hat{\boldsymbol{\chi}}_n - \boldsymbol{\chi}_n^*\|_1 \le \|\hat{\boldsymbol{\chi}}_n\|_1 + \|\boldsymbol{\chi}_n^*\|_1 = 2.$$

Summing over all N nodes yields the global bound

$$\|\hat{\boldsymbol{\chi}} - \boldsymbol{\chi}^*\|_1 = \sum_{n=1}^{N} \|\hat{\boldsymbol{\chi}}_n - \boldsymbol{\chi}_n^*\|_1 \le \sum_{n=1}^{N} 2 = 2N,$$

which provides an explicit worst-case proximity guarantee for the rounding step. □

Here we present the comprehensive complexity analysis for the proposed ADMM algorithm, aligned with the streamlined algorithmic stages. Our reputation-aware ADMM approach

for the MINLP problem achieves polynomial-time complexity through efficient block-structured decomposition. Algorithm 1 exhibits a time complexity of $O(L \times (|\mathcal{J}| + |\mathcal{E}|T + |\mathcal{N}|K))$ for L iterations. The computational complexity analysis corresponds to the algorithm stages as follows: (i) block-structured projection stage (line 4) performs $O(|\mathcal{J}|)$ operations for RSOC projections, $O(|\mathcal{E}|T)$ operations for link load projections, $O(|\mathcal{J}||\mathcal{N}|)$ operations for compute projections, and $O(|\mathcal{N}|K)$ operations for assignment simplex projections, (ii) closed-form proximal operator stage (line 5) requires $O(1)$ operations due to linearity of $g(\mathbf{z}) = -z_\gamma$, and (iii) binary recovery stage (lines 11-12) performs $O((|\mathcal{N}| + K)^3)$ operations using minimum-cost flow for reputation-biased assignment.

The algorithm operates within polynomial-time bounds, ensuring computational efficiency suitable for real-time optimization in resource-constrained multi-tier environments through effective decomposition of the complex MINLP problem. The reputation-guided binary recovery maintains solution feasibility with respect to shard-assignment constraints (C6) and (C8) while incorporating trust-based node selection, providing both theoretical convergence guarantees and practical implementation advantages for distributed blockchain systems.

5.5. Security Analysis

The proposed MTC-SBC framework addresses several critical security challenges inherent to multi-tier computing-enabled sharded blockchain systems. This subsection analyzes the security properties of the system along with their resilience against potential threats.

Byzantine fault tolerance and shard security: MTC-SBC employs BFT-Smart consensus within each shard, which tolerates up to f_k Byzantine nodes in shard k as long as the committee size satisfies $|k| \geq 3f_k + 1$. The reputation-based recruitment mechanism in Eq. (6) ensures that only nodes with reputation $\rho_n(t) \geq \rho^{\text{th}}$ participate as validators, which reduces the likelihood that malicious nodes enter the consensus committee. The total shard reputation ρ_k^{total} aggregates the reputation of all validators in shard k, and the term $-w_2 \sum_{k\in\mathcal{K}} \rho_k^{\text{total}}$ in the objective function steers the optimization toward committees with higher aggregate reputation. In the relaxed objective $\tilde{\mathcal{F}}_0(\cdot)$ used by the ADMM-based solver, this term appears as a linear component in the assignment variables and preserves the preference for high-reputation committees. Under the dense deployment considered in our evaluation, namely a validator pool of 500 peers distributed across up to 50 shards, BFT-Smart is executed *intra-shard* rather than globally. Therefore, the relevant committee is the shard-level validator set rather than the global node population. Under the considered scenarios, shard formation remains sufficiently balanced after reputation-based filtering so that each active shard maintains the minimum committee size required by the underlying BFT condition, and we do not observe any shard violating this requirement in the evaluated scenarios.

Resilience against Sybil and malicious validators: The subjective logic-based reputation model provides inherent protection against Sybil and malicious behavior. Unlike systems where attackers can immediately leverage multiple fake identities, MTC-SBC requires nodes to accumulate positive feedback $F^+_{m\to n}$ through sustained honest behavior before achieving high reputation scores. Newly created Sybil nodes begin with neutral reputation $\rho_{m\to n} = \mu$ and cannot immediately influence shard formation or consensus voting. Constraint (C7) ensures that the average reputation of nodes assigned to each shard remains no smaller than the system-wide baseline ρ^{avg}, while the voting rule in Eq. (7) regulates consensus participation within an assigned shard through the threshold ρ^{th}. For malicious and on-off attackers, the dynamic reputation update in Eq. (1) continuously monitors node behavior through feedback accumulation. Malicious actions increase disbelief $\delta_{m\to n}$ and decrease ρ_n, while the penalty mechanism in Eq. (5) accelerates isolation when $\rho_n(t) < \rho^{\text{th}}$. Eq. (6) defines the eligible validator set, and Eq. (7) assigns zero voting weight to nodes with $\rho_n(t) < \rho^{\text{th}}$, excluding them from consensus participation. The empirical validation in Section 6 demonstrates that malicious nodes are rapidly isolated within 1000 iterations.

Cross-shard transaction integrity and shard isolation: Constraint (C8) in problem **P1** enforces mutually exclusive shard membership, $\sum_k \chi_{n,k} = 1$, which prevents validators from participating in multiple shards simultaneously. This design prevents a single validator from voting in more than one shard at a time and mitigates double-voting attacks across shards, while ensuring that cross-shard transactions are validated by disjoint committees. Combined with the reputation-weighted voting mechanism in Eq. (7), the system makes transaction validity depend on the collective trustworthiness of the validating committee rather than on any individual compromised node. Network slicing further isolates transaction flows associated with different slices and shards. Under these constraints, a local compromise of a subset of nodes affects only the shards where these validators participate, while other shards continue to operate with independent validator sets and reputation dynamics.

In summary, MTC-SBC integrates reputation-based recruitment with BFT consensus to address Byzantine fault tolerance, Sybil resistance, and cross-shard integrity in multi-tier sharded blockchain systems.

6. Performance Evaluation

We evaluated the efficiency and effectiveness of our proposed solution and compared it with current best practices in a sophisticated hybrid architecture, which is split into three layers: (i) the Blockchain layer (i.e., Layer-1) is based on Hyperledger Fabric for distributed ledger management, consensus execution, and smart contract automation; (ii) the Python-based MTC tier (i.e., Layer-2) consists of node emulation for realistic edge–fog–cloud resource heterogeneity and ADMM-based optimization; and (iii) a Python-based API architecture integrates Layers 1 and 2.

6.1. Testbed Implementation

We conducted an experimental evaluation on a high-performance Ubuntu 20.04 LTS server equipped with dual Intel Xeon Gold 6240 processors (including 72 hardware

Table 3: Experiment parameters with tier-specific and global settings.

Tier-specific parameter	**Edge**	**Fog**	**Cloud**
No. of CNs	500	30	2
Computing capacity (GHz/core)	1-3	2-4	4-5
Transmission delay (ms)	5-20	20-50	50-100
Global parameter			
Data forwarding rate (packets/s)	100		
Channel bandwidth (MHz)	80		
Transactions per block (KB)	5-10		
No. of transactions	Up to 10,000		
No. of shards	Up to 50		
No. of tenants	200-1000		
Reputation threshold	0.5		
Task size (KB)	100-500		
No. of iterations	1000		

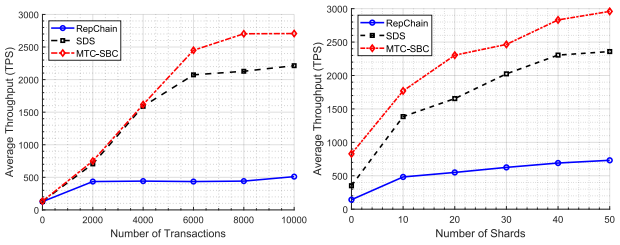
threads at 3.9 GHz boost), 512 GB of memory, a cluster of ten NVIDIA GeForce RTX 2080 Ti GPUs, and NVMe SSD storage. Experimental parameters are presented in Table 3. The entire testbed is containerized using Docker Compose for reproducibility, with deployment configurations and source code publicly available on GitHub[1].

Blockchain network setup (Layer-1): The blockchain layer is emulated using Hyperledger Fabric v2.5[2], deployed via Docker Compose. To accurately capture ledger replication and consensus overhead within the hardware limits of the testbed, we deploy a representative network of 500 peers running in containers. These peers are distributed across the 50 shards (i.e., 10 nodes per shard), satisfying the fault tolerance requirement for the underlying BFT-SMaRt consensus mechanism. Chaincode implements (i) reputation-based shard assignment, (ii) cross-shard transaction routing, and (iii) block verification coordination.

Python-based multi-tier network setup (Layer-2): The multi-tier environment is implemented in Python. Each computing node is instantiated as a separate process with tier membership and heterogeneous computing capacity (1–5 GHz/core). Network behavior is modeled using an asyncio-based event-driven framework with simulated inter-tier latency (10–100 ms) and bandwidth constraints (80 MHz channels). Each node exposes a lightweight Flask REST API for transaction submission, resource allocation requests, and QoS reporting. Network slices are implemented using Linux namespaces, with slice-specific VNFs deployed as containerized services.

Layer Integration: The integration between Layers 1 and 2 is executed through authenticated REST APIs and asynchronous message queues. The Python layer executes the ADMM-based optimization and generates structured outputs, including shard assignments, committee memberships, and resource allocation parameters. These outputs are serialized in JSON format and submitted to Fabric peers via the Fabric SDK gateway.

Baselines and Evaluation Metrics: For comparative analysis, we evaluated the performance of the proposed framework against state-of-the-art baselines selected for their architectural relevance to our specific objectives. To the best of our knowledge, no single contribution has implemented such a complex architecture integrating multi-tier computing, sharding, and network slicing capabilities. Therefore, we selected RepChain [12] and SDS [37] as benchmarks for scalability and consensus metrics, as RepChain's double-chain architecture closely parallels our trust model and SDS specifically optimizes cross-shard transactions. Additionally, to evaluate resource provisioning efficiency, we compared against NetChain [41] and SliceBlock [42], which represent leading blockchain-enabled network slicing solutions. To ensure a fair comparison, we emulated the core consensus logic and communication overheads of these baselines within our unified hybrid testbed. By subjecting all protocols to consistent experimental conditions, including identical network constraints and heterogeneous node capabilities, we ensure that any observed performance differences are attributable to algorithmic efficiency rather than environmental discrepancies. To validate the feasibility of our proposed scheme, we considered multiple metrics for a comprehensive analysis. In the first set of experiments, we performed blockchain and sharding-based analyses in terms of throughput, latency, and reputation score. In the second set of experiments, we conducted MTC and slicing-based analyses to evaluate the performance of our solution in terms of CPU utilization, queueing delay, bandwidth utilization, task completion, and energy efficiency.



(a) (b)

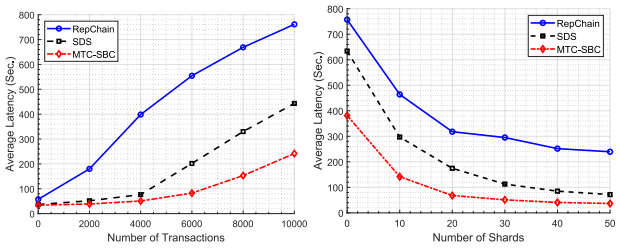
Figure 5: Analysis of throughput against increasing number of (a) transactions, and (b) shards.

6.2. Result Discussion

6.2.1. Analysis of Throughput

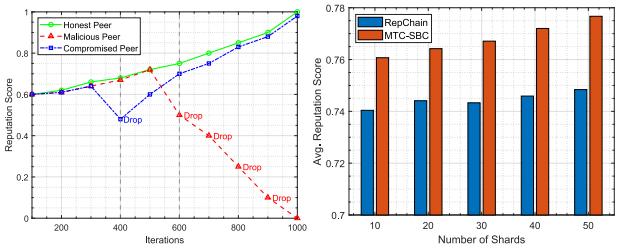
As illustrated in Fig. 5a, the throughput of the proposed MTC-SBC is assessed with a fixed number of 50 shards while increasing the number of blockchain transactions up to 10,000. MTC-SBC shows a notable rise in throughput to 750 TPS with 2,000 transactions, in contrast to SDS's 706 TPS and RepChain's 435 TPS. As the transaction count grows to 4,000, MTC-SBC processes 1,617 TPS, slightly surpassing SDS at 1,590 TPS, while the baseline falls behind at 442 TPS. With 6,000 transactions, MTC-SBC achieves 2,450 TPS, marking a significant improvement over SDS, which reaches 2,073 TPS, and RepChain, which remains at 435 TPS. Progressing to 8,000 transactions, MTC-SBC further increases its throughput to 2,703 TPS, with SDS at 2,128 TPS and RepChain steady at 442 TPS. By 10,000 transactions, MTC-SBC reaches its peak throughput

[1] https://github.com/karimmd/mtc-sbc-exp
[2] https://www.lfdecentralizedtrust.org/projects/fabric



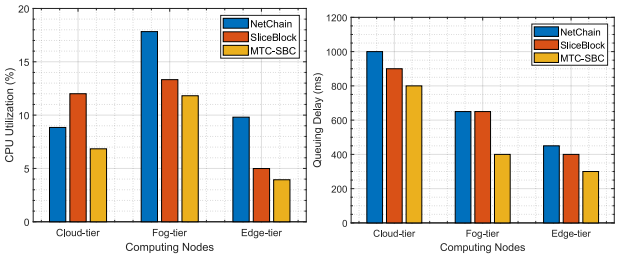
(a) (b)

Figure 6: Analysis of latency against increasing number of (a) transactions, and (b) shards.



(a) (b)

Figure 7: Analysis of reputation score in terms of (a) iterations, and (b) shard numbers.



(a) (b)

Figure 8: Analysis of (a) CPU utilization, and (b) queueing delay across cloud, fog, and edge computing nodes.

of 2,707 TPS, outperforming SDS at 2,213 TPS and RepChain at 511 TPS. This superior performance is attributed to MTC-SBC's service provisioning, which optimizes resource allocation and reduces transaction processing delays.

Figure 5b presents the throughput analysis of MTC-SBC with a fixed 4,000 transactions while varying the number of shards from 0 to 50. At 10 shards, MTC-SBC achieves 1,769 TPS, notably exceeding SDS's 1,385 TPS and RepChain's 482 TPS. Increasing the shard count to 20 results in MTC-SBC's throughput climbing to 2,304 TPS, compared to SDS's 1,654 TPS and RepChain's 550 TPS. With 30 shards, MTC-SBC demonstrates enhanced performance at 2,465 TPS, while SDS achieves 2,025 TPS and RepChain reaches 625 TPS. The throughput of MTC-SBC rises further to 2,831 TPS with 40 shards, in comparison to SDS at 2,305 TPS and RepChain at 691 TPS. At the maximum of 50 shards, MTC-SBC obtains its highest throughput of 2,959 TPS, surpassing SDS at 2,358 TPS and RepChain at 731 TPS. The significant performance improvement with more shards is due to the efficient distribution of computational tasks and minimized inter-shard communication overhead, which collectively enhance the scalability and processing efficiency of the system.

6.2.2. *Analysis of Latency*

As depicted in Figure 6a, we assess the latency of the proposed MTC-SBC with a fixed number of 50 shards while increasing the number of transactions from 0 to 10,000. When the number of transactions reaches 2000, MTC-SBC exhibits an average latency of 38.76 seconds, significantly lower than SDS at 51.68 seconds and RepChain at 179.88 seconds. At 6000 transactions, MTC-SBC's latency is 82.48 seconds, while SDS has a latency of 201.74 seconds and RepChain shows a much higher latency of 554.53 seconds. Finally, at 10,000 transactions, MTC-SBC records the lowest latency at 241.49 seconds, in contrast to SDS's 443.23 seconds and RepChain's 761.24 seconds. The significantly lower latency of MTC-SBC is attributed to its efficient virtual queue management and dynamic resource allocation strategies, which ensure the stability of queue lengths and optimize the processing delay by balancing the computational load across different tiers. These mechanisms reduce bottlenecks and enhance the speed of transaction validation.

As depicted in Fig. 6b, we analyze the latency of the proposed MTC-SBC with a fixed number of 4000 transactions while increasing the number of shards from 0 to 50. When the number of shards is 10, MTC-SBC demonstrates a significant reduction in latency to 141.79 seconds, compared to SDS at 297.04 seconds and RepChain at 464.16 seconds. As the number of shards increases to 20, MTC-SBC's latency further decreases to 67.99 seconds, while SDS achieves 174.65 seconds and RepChain 318.05 seconds. With 40 shards, MTC-SBC records a latency of 40.79 seconds, SDS at 85.10 seconds, and RepChain at 251.74 seconds. Finally, at 50 shards, MTC-SBC achieves the lowest latency of 36.75 seconds, compared to SDS's 71.60 seconds and RepChain's 239.41 seconds. The improved latency performance of MTC-SBC with an increasing number of shards is attributed to its efficient distribution of computational tasks across the shards and minimized inter-shard communication overhead, which enhance the overall scalability and processing efficiency of the system.

6.2.3. *Analysis of Reputation Resilience*

Fig. 7a illustrates the system's resilience against malicious behavior by tracking the reputation evolution of honest, malicious, and compromised peers.

In all experiments, we fix the reputation threshold at $\rho^{\text{th}} = 0.5$, which stays well above the long-run reputation of malicious peers and below that of honest peers. Under the considered attack scenarios, this choice preserves a sufficiently large set of high-reputation validators in each active shard, and we do not observe any shard falling below the minimum committee size required by the underlying BFT consensus. As observed, the reputation of an honest peer steadily converges to 1.0, reflecting consistent protocol compliance. In contrast, the malicious peer's reputation degrades rapidly to 0 within 1,000 iterations, as the subjective logic model in Eq. (1) captures negative

feedback from peer interactions through the belief, disbelief, and indecision parameters. Crucially, this rapid degradation triggers the exclusion mechanism defined in Eq. (6), ensuring that malicious nodes are isolated before they can significantly impact the aggregate shard reputation computed via Eq. (3). Even in the case of a compromised node (simulated using an on-off attack pattern), the system detects the behavioral shift at iteration 400, causing a penalty dip governed by Eq. (5) that recovers only after sustained honest behavior. This individual resilience directly translates to shard-level stability. Fig. 7b demonstrates that MTC-SBC maintains a consistently higher average shard reputation of 0.77 at 50 shards compared to the baseline RepChain, which achieves 0.75 at 50 shards. Since malicious peers rapidly decay toward zero reputation, their contribution to the shard reputation aggregation becomes negligible, preventing them from diluting the collective trustworthiness of the shard. Furthermore, the reputation-weighted voting mechanism, which adjusts voting weight with the exclusion indicator, ensures that even if malicious nodes temporarily infiltrate a shard, their influence is minimized proportionally to their declining reputation scores. This multi-layered defense improves the resilience of the reputation-based recruitment model and helps mitigate Sybil attacks.

6.2.4. *Analysis of CPU Utilization and Queueing Delay*

We compare the CPU utilization (illustrated in Fig. 8a) of our proposed scheme with NetChain and SliceBlock across the cloud, fog, and edge-tiers. In the cloud-tier, SliceBlock exhibits the highest CPU utilization at 12.01%, followed by NetChain at 8.84%, and the proposed MTC-SBC at 6.84%. This indicates that MTC-SBC is more efficient at consuming less CPU power while still maintaining performance. Moving to the fog-tier, NetChain's CPU utilization peaks at 17.83%, which is notably higher than that of both SliceBlock at 13.32% and MTC-SBC at 11.82%. This suggests that overloading resources at this intermediate level leads to inefficiencies. In contrast, MTC-SBC's moderate utilization suggests a balanced approach that provides better scalability and cost-effectiveness in fog computing environments. In the edge-tier, the CPU utilization for all schemes significantly decreases, with NetChain, SliceBlock, and MTC-SBC recording 9.81%, 4.99%, and 3.94%, respectively. The lower CPU utilization of MTC-SBC at the edge reflects its ability to operate efficiently in resource-constrained environments.

In Fig. 8b, we evaluate the queueing delay of our proposed MTC-SBC framework against the baselines. In the cloud-tier, MTC-SBC has a reduced queueing delay at 785 ms, which is a notable improvement over SliceBlock at 853 ms and NetChain at 1014 ms. This indicates that MTC-SBC efficiently manages cloud resources to minimize delays, which is critical for time-sensitive applications. In the case of the fog-tier, MTC-SBC achieves a queueing delay of 631 ms, while outperforming SliceBlock at 719 ms and NetChain at 804 ms. The reduced queueing delay in the fog-tier suggests that our solution handles the allocation of computational resources to prevent bottlenecks and ensure quicker task processing. In the edge-tier, MTC-SBC excels at 378 ms, compared to 435 ms for SliceBlock and 498 ms

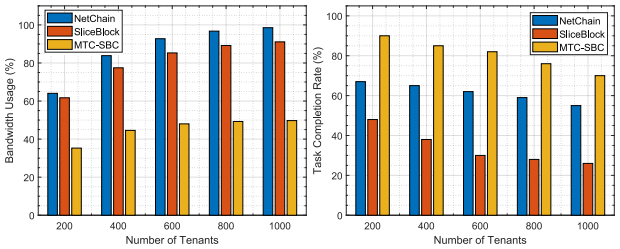


Figure 9: Analysis of (a) bandwidth usage and (b) task completion ratio against the number of tenants.

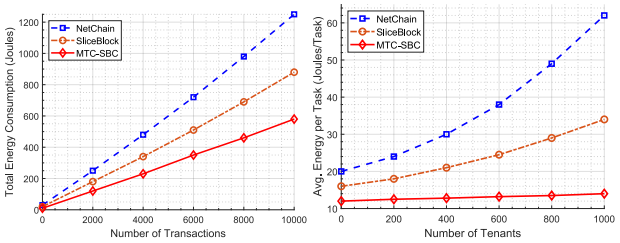


Figure 10: Analysis of (a) energy consumption against increasing transaction loads, and (b) average energy consumption per task.

for NetChain. The reduction in delay underscores MTC-SBC's ability to optimize resources in a constrained environment.

6.2.5. *Analysis of Bandwidth Utilization and Task Completion Ratio*

Fig. 9 illustrates the bandwidth usage and task completion ratio as the number of tenants increases. In Fig. 9a, we observe that MTC-SBC consistently maintains lower bandwidth usage than both SliceBlock and NetChain. For 200 tenants, MTC-SBC uses only 35% of the available bandwidth, while NetChain and SliceBlock utilize 64% and 62%, respectively. As we increase the number of tenants to 1000, MTC-SBC maintains its efficiency, using only 49%, as opposed to NetChain and SliceBlock as they consume more than 90% bandwidth. This demonstrates the ability of our proposed solution to optimize bandwidth usage under increasing loads. In Fig. 9b, it is observed that MTC-SBC significantly outperforms the baseline solutions in terms of task completion ratio. With 200 tenants, MTC-SBC achieves a high rate of 90%, ahead of NetChain's 67% and SliceBlock's 48%. When the number of tenants is increased to 1000, MTC-SBC completes 70% of tasks, while both NetChain and SliceBlock fall to 55% and 26%, respectively. This is due to MTC-SBC's efficient task offloading and reputation-based peer selection, which ensures that only highly reputable nodes are assigned critical tasks in order to minimize the likelihood of incomplete transactions.

6.2.6. *Analysis of Energy Efficiency*

To evaluate the operational sustainability of the proposed framework in power-constrained edge environments, we analyzed the cumulative energy footprint derived from system-level resource logs. Utilizing the standard linear power

consumption model [62], we mapped the real-time resource utilization metrics, specifically the instantaneous CPU load and transceiver bandwidth usage recorded in Fig. 8a and Fig. 9a to dynamic power states. This approach allows for a quantitative estimation of the energy cost associated with protocol execution and consensus overheads. Fig. 10a illustrates the total energy consumption as transaction volume increases, where the MTC-SBC framework demonstrates superior energy efficiency by achieving approximately 34% and 53% lower energy consumption compared to SliceBlock and NetChain, respectively, under maximum load conditions. This performance advantage is attributed to the algorithmic efficiency of the ADMM-based optimization, which significantly reduces the computational duty cycle required for shard allocation. By minimizing the active processing time per transaction block, MTC-SBC reduces the duration that edge nodes spend in high-power states compared to the computationally intensive consensus mechanisms employed by the baselines. Furthermore, Fig. 10b assesses the scalability of the system by profiling the average energy cost per task against tenant density. While baseline solutions exhibit a steep escalation in per-task energy costs due to congestion-induced re-transmissions and prolonged resource contention, MTC-SBC maintains a highly stable profile with only a 12% increase from the lowest to highest tenant density. This stability indicates that MTC-SBC effectively mitigates the excessive power consumption typically associated with high-tenant orchestration, confirming its suitability for deployment in energy-constrained edge environments.

7. Conclusion and Future Work

In this paper, we proposed MTC-SBC, a reputation-based service provisioning scheme for MTC-enabled sharded blockchain system. Our framework integrates edge, fog, and cloud computing resources to create a scalable, efficient, and secure blockchain solution. Through a combination of sharding, network slicing, and reputation-based recruitment, MTC-SBC optimizes resource allocation and minimizes latency, ensuring robust performance even under high transaction loads. Our comprehensive system model, including communication and computation models, and the detailed optimization problem formulation, demonstrate the effectiveness of MTC-SBC in handling complex and demanding applications. Performance evaluations show that MTC-SBC significantly outperforms existing solutions like RepChain and SDS in terms of throughput and latency, highlighting its potential to meet the needs of modern, data-intensive applications.

Looking forward, we envision leveraging advanced AI techniques to further enhance the capabilities of MTC-SBC. Integrating decentralized and generative AI into our framework could enable more intelligent and adaptive resource management to enhance the efficiency of MTC environments in Web3 ecosystem. For instance, decentralized AI can improve decision-making processes across the network, while generative AI models can predict and optimize resource allocation based on dynamic workloads and user demands. By incorporating these AI-driven approaches, we will develop a more resilient, adaptive, and efficient blockchain infrastructure that can seamlessly support the evolving needs of decentralized applications and services in the future.

Acknowledgements

This work was supported by the National Key Research and Development Program of China (No. 2020YFA0909100) and the Basic and Applied Basic Research Foundation of Guangdong Province (No. 2023TQ07A264).

References

[1] Mobile Data Traffic Forecast: Ericsson Mobility Report [online]. (Accessed 27 August 2025).

[2] O. Aouedi, T.-H. Vu, A. Sacco, D. C. Nguyen, K. Piamrat, G. Marchetto, Q.-V. Pham, A survey on intelligent internet of things: Applications, security, privacy, and future directions, IEEE Communications Surveys & Tutorials 27 (2) (2025) 1238–1292.

[3] K. Wang, J. Jin, Y. Yang, T. Zhang, A. Nallanathan, C. Tellambura, B. Jabbari, Task Offloading With Multi-Tier Computing Resources in Next Generation Wireless Networks, IEEE Journal on Selected Areas in Communications 41 (2) (2023) 306–319.

[4] A. Islam, S. Chakraborty, M. Ghose, Remec: Reliability-aware scheduling of mixed-criticality iot tasks in dvfs-enabled multi-tier edge computing, Future Generation Computer Systems (2025) 108074.

[5] Y. Wang, N. Li, P. Yu, W. Li, X. Qiu, S. Wang, M. Cheriet, Intelligent and collaborative orchestration of network slices, IEEE Transactions on Services Computing 16 (2) (2022) 1239–1253.

[6] P. Wang, H. Li, H. Fu, Z. Sun, J. Chen, X. Du, A blockchain system for qos monitoring in decentralized edge computing, IEEE Transactions on Services Computing 17 (1) (2024) 263–276.

[7] M. M. Karim, Q. Qu, Y. Cai, T. Liu, X. Meng, Bitcoin reimagined: A comprehensive study of ordinals and inscriptions protocols for web3 asset innovation, Blockchain: Research and Applications (2025) 100379.

[8] G. Wang, Repshard: reputation-based sharding scheme achieves linearly scaling efficiency and security simultaneously, in: 2020 IEEE International Conference on Blockchain, IEEE, 2020, pp. 237–246.

[9] A. A. Battah, Y. Iraqi, E. Damiani, A trust and reputation system for iot service interactions, IEEE Transactions on Network and Service Management 19 (3) (2022) 2987–3005.

[10] M. Zamani, M. Movahedi, M. Raykova, Rapidchain: Scaling blockchain via full sharding, in: ACM conference on computer and communications security, 2018, pp. 931–948.

[11] H. Liu, B. Ma, Y. Chen, J. Zhang, B. Liu, H. Chen, D. Deng, Blockchain assisted secure embedding of virtual networks in multi-domain elastic optical network, IEEE Transactions on Network and Service Management 22 (5) (2025) 3838–3848.

[12] C. Huang, Z. Wang, H. Chen, Q. Hu, Q. Zhang, W. Wang, X. Guan, RepChain: A Reputation-Based Secure, Fast, and High Incentive Blockchain System via Sharding, IEEE Internet of Things Journal 8 (6) (2021) 4291–4304.

[13] M. Deng, Y. Lyu, C. Yang, F. Xu, M. Ahmed, N. Yang, Z. Xu, C. Ke, Lightweight trust management scheme based on blockchain in resource-constrained intelligent iot systems, IEEE Internet of Things Journal 11 (15) (2024) 25706–25719.

[14] N. Sharghivand, L. Mashayekhy, W. Ma, S. Dustdar, Time-constrained service handoff for mobile edge computing in 5g, IEEE Transactions on Services Computing 16 (3) (2022) 2241–2253.

[15] C. Gonçalves, J. Simão, L. Veiga, A function-as-a-service middleware for decentralized collaborative edge computing, Future Generation Computer Systems (2025) 108069.

[16] C. Sun, X. Li, C. Wang, Q. He, X. Wang, V. C. M. Leung, Hierarchical deep reinforcement learning for joint service caching and computation offloading in mobile edge-cloud computing, IEEE Transactions on Services Computing 17 (4) (2024) 1548–1564.

[17] M. M. Salim, M. Kim, S. K. Singh, J. H. Park, Zero-trust blockchain-enabled framework for scalable and secure iot networks, Future Generation Computer Systems (2025) 108093.
[18] G. Xu, Z. Zhou, X. Song, Y. Huang, Research on transaction allocation strategy in blockchain state sharding, Future Generation Computer Systems 168 (2025) 107756.
[19] E.-h. Diallo, K. Al Agha, S. Martin, Trade-5g: A blockchain-based transparent and secure resource exchange for 5g network slicing, Blockchain: Research and Applications 6 (1) (2025) 100246.
[20] A. K. Yadav, S. Wijethilaka, M. Liyanage, Blockchain-based cross-operator network slice authentication protocol for 5g communication, IEEE Transactions on Network and Service Management 22 (4) (2025) 3106–3119.
[21] F. Gai, B. Wang, W. Deng, W. Peng, Proof of reputation: A reputation-based consensus protocol for peer-to-peer network, in: Database Systems for Advanced Applications, Springer, 2018, pp. 666–681.
[22] R. Di Pietro, X. Salleras, M. Signorini, E. Waisbard, A blockchain-based Trust System for the Internet of Things, in: ACM on Access Control Models and Technologies, New York, NY, USA, 2018, pp. 77–83.
[23] J. Kang, Z. Xiong, D. Niyato, D. Ye, D. I. Kim, J. Zhao, Toward Secure Blockchain-Enabled Internet of Vehicles: Optimizing Consensus Management Using Reputation and Contract Theory, IEEE Transactions on Vehicular Technology 68 (3) (2019) 2906–2920.
[24] S. Lee, S.-H. Seo, Design of a Two Layered Blockchain-Based Reputation System in Vehicular Networks, IEEE Transactions on Vehicular Technology 71 (2) (2022) 1209–1223.
[25] X. Wu, Z. Wang, X. Li, L. Chen, Dbpbft: A hierarchical pbft consensus algorithm with dual blockchain for iot, Future Generation Computer Systems 162 (2025) 107429.
[26] E. K. Wang, Z. Liang, C.-M. Chen, S. Kumari, M. K. Khan, PoRX: A reputation incentive scheme for blockchain consensus of IIoT, Future Generation Computer Systems 102 (2020) 140–151.
[27] W. Zhao, X. Yang, S. Qi, J. Wei, X. Dong, Y. Qi, Secure blockchain-based reputation system for iiot-enabled retail industry with resistance to sybil attack, Future Generation Computer Systems 166 (2025) 107705.
[28] J. Yu, D. Kozhaya, J. Decouchant, P. Esteves-Verissimo, RepuCoin: Your Reputation Is Your Power, IEEE Transactions on Computers 68 (8) (2019) 1225–1237.
[29] S. Li, W. Ma, Efficient complex task offloading in blockchain-secured edge computing using a many-objective evolutionary algorithm, Future Generation Computer Systems 174 (2026) 107937.
[30] Y. Cao, C. Zhao, Y. Zhang, Y. Jin, Optimizing resource allocation and energy efficiency in vehicle mobile edge computing with blockchain integration, IEEE Internet of Things Journal 12 (18) (2025) 36807–36818.
[31] J. Zheng, Y. Zhang, Trbft: An efficient blockchain consensus for edge computing-enabled iot systems, IEEE Internet of Things Journal 12 (11) (2025) 15853–15868.
[32] E. Kokoris-Kogias, P. Jovanovic, L. Gasser, N. Gailly, E. Syta, B. Ford, OmniLedger: A Secure, Scale-Out, Decentralized Ledger via Sharding, in: IEEE Symposium on Security and Privacy, 2018, pp. 583–598.
[33] M. H. Manshaei, M. Jadliwala, A. Maiti, M. Fooladgar, A Game-Theoretic Analysis of Shard-Based Permissionless Blockchains, IEEE Access 6 (2018) 78100–78112.
[34] J. Wang, H. Wang, Monoxide: Scale out blockchains with asynchronous consensus zones, in: USENIX symposium on networked systems design and implementation, 2019, pp. 95–112.
[35] Z. Hong, S. Guo, P. Li, W. Chen, Pyramid: A Layered Sharding Blockchain System, in: IEEE INFOCOM 2021 - IEEE Conference on Computer Communications, 2021, pp. 1–10.
[36] S. Li, M. Yu, C.-S. Yang, A. S. Avestimehr, S. Kannan, P. Viswanath, PolyShard: Coded Sharding Achieves Linearly Scaling Efficiency and Security Simultaneously, IEEE Transactions on Information Forensics and Security 16 (2021) 249–261.
[37] S. K. Set, G. S. Park, Service-Aware Dynamic Sharding Approach for Scalable Blockchain, IEEE Transactions on Services Computing 16 (4) (2023) 2954–2969.
[38] Z. Xu, A. S. Shahraki, N. Chilamkurti, A fault-tolerant sharding mechanism for resilience and scalability in blockchain using backup pool, Future Generation Computer Systems 174 (2026) 107982.
[39] X. Xiong, M. Li, F. R. Yu, H. Zhang, K. Wang, P. Si, Cloud-edge-end collaborative computing-enabled intelligent sharding blockchain for industrial iot based on ppo approach, IEEE Transactions on Mobile Computing 24 (9) (2025) 8011–8024.
[40] E. Bandara, S. Shetty, A. Rahman, R. Mukkamala, X. Liang, Moose: A Scalable Blockchain Architecture for 5G Enabled IoT with Sharding and Network Slicing, in: IEEE Wireless Communications and Networking Conference, 2022, pp. 1194–1199.
[41] G. He, W. Su, S. Gao, N. Liu, S. K. Das, NetChain: A Blockchain-Enabled Privacy-Preserving Multi-Domain Network Slice Orchestration Architecture, IEEE Transactions on Network and Service Management 19 (1) (2022) 188–202.
[42] I. H. Abdulqadder, S. Zhou, Sliceblock: Context-aware authentication handover and secure network slicing using dag-blockchain in edge-assisted sdn/nfv-6g environment, IEEE Internet of Things Journal 9 (18) (2022) 18079–18097.
[43] L. M. Garcia-Saez, S. Ruiz-Villafranca, J. Roldan-Gomez, J. Carrillo-Mondejar, J. L. Martinez, Poisoning-resilient federated learning for mec-iot environments using blockchain, ACM Transactions on Internet Technology (2025).
[44] F. Javed, J. Mangues-Bafalluy, E. Zeydan, L. Blanco, Trustworthy reputation for federated learning in o-ran using blockchain and smart contracts, IEEE Open Journal of the Communications Society 6 (2025) 1343–1362.
[45] P. Liu, M. Zhang, Z. Shen, H. Wang, Federated learning with privacy protection and incentive mechanisms based on edge computing, Future Generation Computer Systems (2025) 108216.
[46] A. Bessani, J. Sousa, E. E. Alchieri, State machine replication for the masses with bft-smart, in: 2014 44th Annual IEEE/IFIP International Conference on Dependable Systems and Networks, IEEE, 2014, pp. 355–362.
[47] T. Kwantwi, G. Sun, N. A. E. Kuadey, G. T. Maale, G. Liu, Blockchain-based computing resource trading in autonomous multi-access edge network slicing: A dueling double deep q-learning approach, IEEE Transactions on Network and Service Management 20 (3) (2023) 2912–2928.
[48] E. Zeydan, J. Baranda, J. Mangues-Bafalluy, Y. Turk, S. B. Ozturk, Blockchain-based service orchestration for 5g vertical industries in multicloud environment, IEEE Transactions on Network and Service Management 19 (4) (2022) 4888–4904.
[49] T. Hewa, P. Porambage, I. Kovacevic, N. Weerasinghe, E. Harjula, M. Liyanage, M. Ylianttila, Blockchain-based network slice broker to facilitate factory-as-a-service, IEEE Transactions on Industrial Informatics 19 (1) (2022) 519–530.
[50] G. O. Boateng, D. Ayepah-Mensah, D. M. Doe, A. Mohammed, G. Sun, G. Liu, Blockchain-Enabled Resource Trading and Deep Reinforcement Learning-Based Autonomous RAN Slicing in 5G, IEEE Transactions on Network and Service Management 19 (1) (2022) 216–227.
[51] G. O. Boateng, G. Sun, D. A. Mensah, D. M. Doe, R. Ou, G. Liu, Consortium Blockchain-Based Spectrum Trading for Network Slicing in 5G RAN: A Multi-Agent Deep Reinforcement Learning Approach, IEEE Transactions on Mobile Computing (2022) 1–15.
[52] Y. Gong, S. Sun, Y. Wei, M. Song, Deep reinforcement learning for edge computing resource allocation in blockchain network slicing broker framework, in: IEEE Vehicular Technology Conference, IEEE, 2021, pp. 1–6.
[53] P. Gorla, V. Chamola, V. Hassija, D. Niyato, Network Slicing for 5G with UE State Based Allocation and Blockchain Approach, IEEE Network 35 (3) (2021) 184–190.
[54] A. K. Yadav, S. Wijethilaka, A. Braeken, M. Misra, M. Liyanage, An enhanced cross-network-slice authentication protocol for 5g, IEEE Transactions on Sustainable Computing 8 (4) (2023) 555–573.
[55] X. Luo, K. Xue, J. Li, R. Li, D. S. Wei, Make rental reliable: Blockchain-based network slice management framework with sla guarantee, IEEE Communications Magazine 61 (7) (2023) 142–148.
[56] S. Bukhari, K. Sharif, L. Zhu, C. Xu, F. Li, S. Biswas, Dynamic fine-grained sla management for 6g embb-plus slice using mdnn & smart contracts, IEEE Transactions on Services Computing 17 (6) (2024) 3499–3512.
[57] N. M. Chacko, N. VG, M. Balachandra, M. T, Lightweight consensus in blockchain: A systematic literature review, ACM Computing Surveys 58 (3) (2025) 1–37.
[58] F. Javed, K. Antevski, J. Mangues-Bafalluy, L. Giupponi, C. J. Bernardos,

Distributed Ledger Technologies for Network Slicing: A Survey, IEEE Access 10 (2022) 19412–19442.
[59] G. Liu, Z. Wu, Y. Zhou, Y. Liu, H. Kang, Communitychain: Toward a scalable blockchain in smart home, IEEE Transactions on Network and Service Management 20 (3) (2023) 2898–2911.
[60] 3GPP, Study on management and orchestration of network slicing for next generation network, 3GPP TR 28.801 V15.1.0 (2018).
[61] M. Alam, N. Ahmed, S. Ghosh, R. Matam, F. A. Barbhuiya, Optifog: A framework for acquiring state information and predicting resource availability for task offloading in cooperative fog-networks, IEEE Transactions on Services Computing (2024) 1–13.
[62] X. Fan, W.-D. Weber, L. A. Barroso, Power provisioning for a warehouse-sized computer, in: Proceedings of the 34th Annual International Symposium on Computer Architecture (ISCA), ACM, 2007, pp. 13–23.