

Vadym Y. Mukhin

Professor of department of the mathematical methods of system analysis of National Technical University of Ukraine "Kiev Polytechnic Institute", Doct. of Sc.

ORCID: 0000-0003-3833-1529

v_mukhin@i.ua

Viacheslav O. Kulyk

Assistant at the Department of System Design of the National Technical University of Ukraine "Kyiv Polytechnic Institute"

ORCID: 0000-0003-3833-1529

slava.kulik27@gmail.com

GOSSIP-BASED TRUST AND REPUTATION MECHANISM FOR BROKER INTERACTION IN NETWORK-CENTRIC DISTRIBUTED COMPUTER SYSTEMS

Abstract. The object of this research is the broker interaction layer of Network-Centric Distributed Computer Systems (DCS), in which autonomous peer-to-peer brokers replace centralized control planes to manage resource allocation and task scheduling. The problem addressed is the absence of a decentralized trust mechanism capable of detecting and isolating malicious or Byzantine brokers (nodes that silently drop tasks, falsify resource availability, or engage in collusion) without reintroducing the single points of failure that the network-centric architecture was designed to eliminate. To resolve this, a gossip-based trust and reputation mechanism is proposed and integrated directly into the NCA broker interaction layer. The mechanism introduces a composite reputation vector comprising three task-centric metrics (Task Success Rate, Task Delay Factor, and Fault Tolerance Score), coupled with an exponential temporal decay function to rapidly detect oscillating malicious behavior. A credibility-weighted reconciliation rule neutralizes localized collusion by weighting incoming gossip reports against the reporter's own trust score. These trust states serve as a zero-latency pre-selection filter for the task scheduler. The results are explained by the logarithmic convergence of the push-pull epidemic protocol, which propagates trust updates across all honest brokers in $O(\log N)$ rounds without centralized coordination. Empirical simulations confirm that the mechanism sustains over 93% task completion rates under 20% Byzantine fault injection and reduces makespan by approximately 25%, generating fewer than two additional gossip messages per scheduled task. The mechanism can be applied in any distributed computing environment operating a horizontal broker control plane, particularly in edge computing, IoT, and cloud federation scenarios where centralized security authorities are impractical.

Keywords: Distributed Computer Systems, Network-Centric Architecture, Gossip Protocol, Trust and Reputation Management, Byzantine Fault Tolerance, Broker Interaction, Epidemic Dissemination

Мухін Вадим Євгенович

Професор кафедри математичних методів системного аналізу Національного технічного університету України «Київський політехнічний інститут», доктор наук

ORCID: 0000-0003-3833-1529

v_mukhin@i.ua

Кулик Вячеслав Олександрович

Асистент кафедри системного проектування Національного технічного університету України «Київський політехнічний інститут»

ORCID: 0000-0003-3833-1529

slava.kulik27@gmail.com

МЕХАНІЗМ ДОВІРИ ТА РЕПУТАЦІЇ НА ОСНОВІ ПЛІТОК ДЛЯ ВЗАЄМОДІЇ БРОКЕРІВ У МЕРЕЖЕВО-ОРІЄНТОВАНИХ РОЗПОДІЛЕНИХ КОМП'ЮТЕРНИХ СИСТЕМАХ

Анотація. Предметом даного дослідження є рівень взаємодії брокерів у мережево-орієнтованих розподілених комп'ютерних системах (DCS), в яких автономні брокери типу «рівний-рівному» замінюють централізовані рівні управління для керування розподілом ресурсів та плануванням завдань. Проблема, що розглядається, полягає у відсутності децентралізованого механізму довіри, здатного виявляти та ізолювати зловмисних або візантійських брокерів (вузлів, які непомітно відкидають завдання, фальсифікують доступність ресурсів або вступають у змову) без повторного введення одиничних точок відмови, які мережево-орієнтована архітектура була покликана усунути. Для вирішення цієї проблеми пропонується механізм довіри та репутації на основі «пліток», який інтегровано безпосередньо в рівень взаємодії брокерів NCA. Цей механізм запроваджує комбінований вектор репутації, що складається з трьох показників, орієнтованих на завдання (коефіцієнт успішності завдань,

© 2026 V. Mukhin, V. Kulyk. This work is licensed under CC BY 4.0.

<https://creativecommons.org/licenses/by/4.0/>

коефіцієнт затримки завдань та показник відмовостійкості), у поєднанні з експоненціальною функцією часового згасання для швидкого виявлення коливальної зловмисної поведінки. Правило узгодження, зважене за надійністю, нейтралізує локалізовану змову шляхом зважування вхідних повідомлень «gossip» відповідно до власного показника довіри того, хто їх надсилає. Ці стани довіри слугують фільтром попереднього відбору з нульовою затримкою для планувальника завдань. Результати пояснюються логарифмічною збіжністю епідемічного протоколу «push-pull», який поширює оновлення довіри серед усіх чесних брокерів за $O(\log N)$ раундів без централізованої координації. Емпіричні симуляції підтверджують, що механізм забезпечує рівень виконання завдань понад 93 % за умови 20 % введення візантійських збоїв та скорочує загальний час виконання приблизно на 25 %, генеруючи менше ніж два додаткові повідомлення типу «gossip» на кожне заплановане завдання. Механізм може застосовуватися в будь-якому розподіленому обчислювальному середовищі, що використовує горизонтальну площину керування посередниками, зокрема в сценаріях периферійних обчислень, Інтернету речей (IoT) та хмарних федерацій, де централізовані органи безпеки є недоцільними.

Ключові слова: розподілені комп'ютерні системи, мережево-орієнтована архітектура, протокол «Gossip», управління довірою та репутацією, стійкість до візантійських відмов, взаємодія з брокерами, епідемічне поширення

1. Introduction

1.1 Background

The structural evolution of Distributed Computer Systems (DCS) has progressively shifted away from rigid, hierarchical resource management paradigms toward dynamic, horizontal topologies. A foundational advancement in this domain is the mandatory base paper by Hu, Mukhin, Kornaga, Lavrenko, and Herasymenko (2017) titled "Distributed Computer System Resources Control Mechanism Based on Network-Centric Approach" [7]. In this framework, centralized control planes are replaced by a distributed layer of autonomous brokers that interact peer-to-peer to manage resource allocation, task scheduling, and load balancing. This NCA approach significantly enhances horizontal scalability and operational agility, enabling the system to process complex, distributed workloads across geographically dispersed computing nodes without relying on a singular point of coordination. The transition to this architecture reflects a broader industry trend toward edge computing, Internet of Things (IoT) integration, and decentralized resource pooling. In these modern paradigms, the sheer volume of computational requests and the geographical distribution of data sources render centralized master-slave configurations obsolete. Centralized schedulers inevitably become severe bandwidth and processing bottlenecks, suffering from high latency and single points of failure. By distributing the scheduling logic across a horizontal plane of peer brokers, the NCA model achieves superior fault tolerance against localized network outages and provides the elasticity required to handle highly variable, bursty workloads.

1.2 Problem statement

While the NCA broker model resolves traditional bottlenecks associated with centralized scheduling, it inherently introduces severe security and operational vulnerabilities. In the absence of a global authoritative entity, brokers operate as critical, largely unmonitored intermediaries. This lack of oversight allows compromised, selfish, or faulty brokers to degrade system performance through various misbehaviors. Such misbehaviors include silent task dropping (omission faults), excessive execution delays (timing faults), and the falsification of available computational capacity (commission faults). Furthermore, rational workers may behave selfishly to minimize their own workloads while exploiting network resources [9], whereas Byzantine adversaries may actively collude to subvert network consensus, disrupt task execution, or launch sophisticated data poisoning attacks against the scheduling logic [10]. The fundamental problem lies in the absence of a decentralized trust mechanism capable of detecting and isolating these malicious actors without reintroducing the single points of failure that the NCA was designed to eliminate. Traditional security models rely heavily on static authentication and cryptographic access controls, which fail to address dynamic behavioral degradation or insider threats where previously authenticated nodes begin acting maliciously due to compromise or rational selfishness.

1.3 Motivation

The motivation for addressing this vulnerability stems from the proven success of epidemic information dissemination in unstructured peer-to-peer networks. Decentralized reputation systems leveraging gossip-based protocols have demonstrated exceptional efficacy in isolating malicious actors and mitigating the free-rider problem [19][12]. These protocols propagate state information through random peer exchanges, achieving convergence in logarithmic time while maintaining self-organizing resilience. Applying this probabilistic methodology directly to the DCS broker layer presents an unprecedented opportunity to bridge decentralized resource control with robust security. By enabling brokers to continuously evaluate and share performance data regarding their peers, the architecture can autonomously maintain a resilient trust overlay without relying on computationally heavy global consensus algorithms or centralized certificate authorities. This ensures that the high-throughput, low-latency requirements of distributed task scheduling are preserved while simultaneously fortifying the network against Byzantine interference.

1.4 Research Aim and Objectives

The primary aim of this research is to conceptualize, formalize, and empirically evaluate a gossip-based trust and reputation mechanism tailored specifically as an extension to the NCA broker model. To achieve this aim, the study pursues the following specific objectives:

1. To formalize a multi-dimensional reputation model that quantifies broker reliability through metrics such as task completion latency, resource reporting accuracy, and historical interaction success.

2. To design a high-efficiency, push-pull epidemic protocol capable of propagating trust vectors across the broker overlay while minimizing the communication overhead inherent in large-scale DCS.

3. To develop a trust-weighted broker selection algorithm that integrates with the existing routing logic to dynamically isolate malicious or underperforming nodes.

4. To rigorously evaluate the proposed mechanism under varying concentrations of Byzantine fault injection to quantify its impact on task completion rates and system makespan.

1.5 Scientific Contributions

Novel contributions of the proposed trust and reputation model are:

- First, this research represents the unprecedented, direct integration of a push-pull gossip trust overlay into the specifically defined NCA broker model [7]. It transforms the previously vulnerable horizontal control plane into a self-securing, Byzantine fault-tolerant network.
- Second, the formulation introduces an innovative multi-dimensional trust metric strictly evaluating task-centric parameters (Task Success Rate, Task Delay Factor, Fault Tolerance Score). This is coupled with an exponential temporal decay function designed specifically to rapidly detect and penalize high-frequency oscillating malicious behavior, a vulnerability often ignored in static reputation systems.
- Third, the methodology extends standard anti-entropy gossip by introducing an original reconciliation rule that weights conflicting reputation reports based on the reporter's own dynamic trust score. This mathematical approach effectively neutralizes localized collusion and false-reporting attacks without requiring global consensus.
- Finally, the approach proposes a distinct algorithmic integration that modifies existing distributed computer system scheduler logic, utilizing dynamically updated gossip states to exclude malicious nodes prior to load balancing, providing a highly reliable routing mechanism without the overhead of continuous global evaluations.

1.6 Paper Organization

The remainder of this paper is organized as follows. Section 2 provides a comprehensive analysis of related work, examining existing trust models, gossip protocols, and security threats in distributed architectures. Section 3 formalizes the problem statement, detailing the NCA broker architecture and defining broker misbehavior. Section 4 presents the proposed trust and reputation model, including the mathematical formulation of reputation metrics and the gossip dissemination protocol. Section 5 conducts a rigorous security analysis against common attack vectors such as false reporting, collusion, and Sybil attacks, alongside an evaluation of security overhead. Section 6 details the experimental setup and discusses the empirical findings regarding convergence, task completion rates, resource utilization, and communication overhead. Finally, Section 7 summarizes the key contributions, highlights experimental outcomes, and outlines directions for future research.

2. Analysis of Related Work

2.1 Trust and Reputation Models in Distributed Systems

Foundational trust models such as EigenTrust [23] and PeerTrust [22] aggregate local peer observations into global reputation scores using recursive power-iteration or transaction-context weighting. Later refinements, including EigenTrust++ [14], added feedback credibility modeling. Probabilistic approaches such as the Beta Reputation System and Subjective Logic apply Bayesian inference to model future node behavior, but impose significant computational overhead unsuitable for latency-sensitive scheduling. Blockchain-based reputation tracking [9] guarantees immutability at the cost of consensus latency, while machine-learning and federated trust models [9] introduce model-poisoning vulnerabilities and convergence times spanning hours. Hardware-assisted mechanisms relying on Trusted Execution Environments (Intel SGX, ARM TrustZone) offer cryptographic integrity guarantees but suffer from vendor lock-in and side-channel vulnerabilities in heterogeneous deployments. Across all these approaches, the recurring limitation is the inability to deliver low-latency, localized trust assessments compatible with millisecond-scale routing decisions in NCA task scheduling.

2.2 Gossip Protocols in Distributed and P2P Systems

Epidemic protocols achieve network-wide state convergence in $O(\log N)$ communication rounds by propagating updates through randomized peer exchanges, maintaining resilience under node failures and network partitions [18]. GossipTrust [19] demonstrated fast reputation aggregation in unstructured peer-to-peer networks, while differential gossip algorithms [12] further reduce communication complexity. Advanced variants such as the Push-Sum algorithm support decentralized aggregation, and anti-entropy gossip underpins eventual consistency in systems such as Apache Cassandra. However, existing database-centric anti-entropy protocols assume static, trusted nodes and are insufficient for volatile, potentially malicious broker environments. Adapting gossip for trust dissemination in NCA broker overlays requires augmentation with credibility-weighted reconciliation to prevent malicious actors from poisoning reputation states during synchronization.

2.3 Security Threats in Broker-Based Distributed Architectures

Autonomous broker intermediaries are exposed to a range of attacks in the absence of centralized oversight [13]. Selfish rational workers minimize their own workloads while exploiting network resources [9]; Byzantine adversaries drop tasks, falsify results, or manipulate resource metrics [10]; and Sybil attackers generate fictitious identities to dilute negative reputation scores [3]. Colluding broker rings further compound these threats by mutually inflating reputations while downgrading honest peers. Existing defenses frequently rely on heavyweight cryptographic consensus protocols that introduce latency incompatible with high-throughput task scheduling requirements.

2.4 Gap Analysis

Despite independent progress in trust modeling and epidemic dissemination, no existing methodology directly applies a lightweight, push-pull gossip trust overlay to the specific NCA broker model defined in [7]. Current solutions either impose unacceptable latency through global consensus, rely on computationally prohibitive models, or fail to capture the behavioral nuances of task-scheduling intermediaries. The proposed mechanism addresses this gap by tailoring epidemic reputation propagation to the operational parameters of horizontal broker networks, delivering eventually consistent trust states without compromising real-time scheduling efficiency.

3. Problem Statement

3.1 NCA Broker Architecture

The NCA distributed computer system (DCS) formalized by Hu et al. [7] operates across three distinct logical layers: the Application Layer, the Broker Layer, and the Resource Layer. The application layer consists of independent schedulers responsible for generating computational task batches, mathematically denoted as the set:

$$S = \{s_1, s_2, \dots, s_m\} \#(1)$$

These schedulers forward resource requests to the horizontal control plane, which comprises a decentralized network of autonomous routing brokers:

$$B = \{b_1, b_2, \dots, b_n\} \#(2)$$

The broker layer can be formally modeled as an unstructured peer-to-peer overlay network, represented by a directed graph $G = (B, E)$, where B is the set of brokers and E represents the active communication links between them. Each edge $e_{i,j} \in E$ is characterized by a dynamic latency $l_{i,j}$ and bandwidth capacity $c_{i,j}$. Brokers dynamically map these incoming tasks to available computational nodes within the physical resource layer, defined as $R = \{r_1, r_2, \dots, r_k\}$. As illustrated in Fig. 1, this decentralized routing architecture eliminates the single point of failure inherent in traditional master-slave scheduling paradigms. The mapping function executed by the brokers must continuously evaluate the state of R to ensure optimal load balancing.

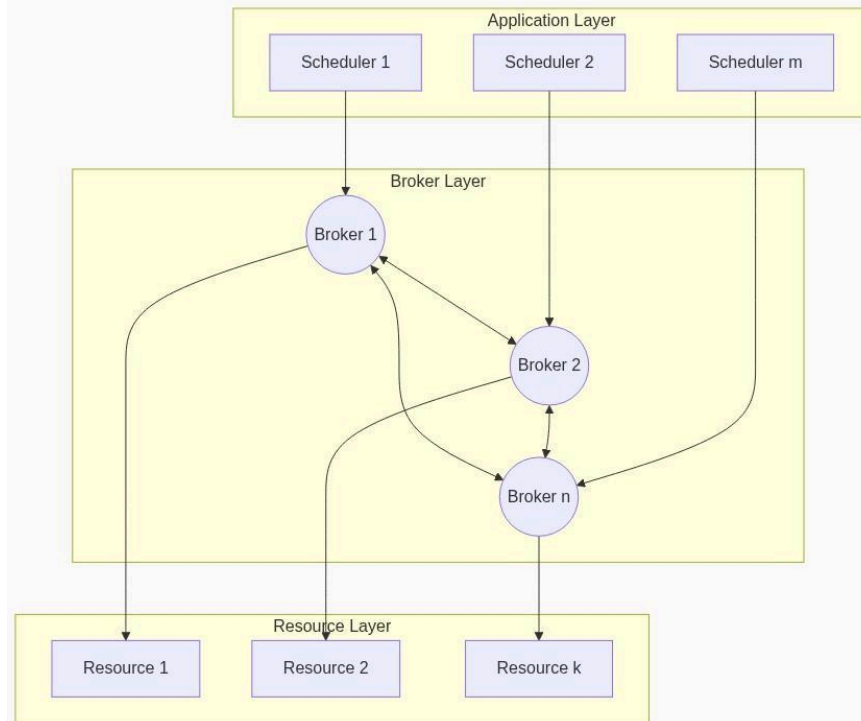


Fig. 1. Diagram illustrating the three-tier network-centric broker architecture with application, broker, and resource layers.

The formalization of this architecture reveals a critical dependency on the cooperative integrity of the broker set B . In the original framework [7], the absence of a verification layer implies that any node b_i is assumed to act in the best interest of the system. Consequently, the problem addressed herein is the formalization of trust metrics and dissemination

protocols required to maintain system stability when this assumption of universal cooperation is violated by Byzantine or selfish actors within the horizontal control plane.

3.2 Formal Definition of Broker Misbehavior

Broker misbehavior within this architecture manifests when a node $b_i \in B$ fails to honestly, securely, or efficiently process a delegated task τ . Let $t_{expected}$ represent the optimal execution time based on resource availability, and t_{actual} represent the observed completion time (see Table 1). A misbehavior function, $M(b_i, \tau)$, evaluates the discrepancy between the expected and actual states:

$$M(b_i, \tau) = \{1, \text{ if task is dropped or falsified } \frac{t_{actual} - t_{expected}}{t_{expected}}, \text{ if } t_{actual} > t_{expected} \quad 0, \text{ otherwise } \#(3)$$

Byzantine or selfish brokers [9] frequently exhibit oscillating behavior, such as on-off attacks [9], alternating between honest execution ($M = 0$) and malicious actions ($M \geq 1$) to evade static detection thresholds. To capture these nuances, the proposed mechanism employs broker-specific composite reputation metrics combined with an exponential temporal decay function to rapidly detect and penalize high-frequency behavioral oscillation. The mathematical representation of this misbehavior ensures that both absolute failures (dropped tasks) and relative failures (intentional delays) are quantitatively captured by the observing nodes. Beyond simple omission and timing faults, commission faults represent a particularly insidious class of misbehavior in decentralized architectures. In a commission fault, a malicious broker actively falsifies its available computational capacity to attract a disproportionate volume of task assignments, a strategy often employed in data poisoning or black-hole attacks. By advertising artificially high MIPS (Millions of Instructions Per Second) capabilities, the compromised broker subverts the load-balancing logic of honest peers, causing them to route critical workloads into a controlled bottleneck. Once the tasks are captured, the adversary may either drop them entirely or execute them using manipulated datasets to return invalid results. Detecting these commission faults requires a trust mechanism that not only monitors binary task completion but also cross-references advertised resource availability against historical execution latency. The multi-dimensional misbehavior function $M(b_i, \tau)$ is specifically designed to capture these discrepancies, ensuring that brokers who consistently over-promise and under-deliver are rapidly identified and penalized by the reputation overlay.

Table 1

Classification of broker misbehavior types			
Misbehavior Category	Mathematical Condition	System Impact	Detection Metric
Task Dropping / Falsification	$M(b_i, \tau) = 1$	Task fails to complete or results are invalid	Task Success Rate (TSR)
Execution Delay	$M(b_i, \tau) = \frac{t_{actual}}{t_{ex}}$	Increases task completion latency	Task Delay Factor (TDF)
Resource Hoarding (Selfish)	Refusal to accept redirected tasks	Degrades load balancing and horizontal sharing	Fault Tolerance Score (FTS)
On-Off Attack (Oscillating)	Alternating $M = 0$ and $M \geq 1$	Evades static detection thresholds	Temporal Decay Function

3.3 Requirements for the Trust Mechanism

To effectively secure the broker layer without degrading the inherent efficiency of the NCA DCS, the proposed trust mechanism must satisfy stringent operational constraints. First, it must exhibit high probabilistic resilience, maintaining near-optimal task success rates even under severe fault injection. Second, the mechanism requires strictly bounded communication overhead, necessitating an epidemic dissemination protocol rather than computationally heavy global consensus. Third, the system must actively resist localized collusion, where groups of malicious brokers artificially inflate peer reputations through coordinated misinformation. This necessitates a robust mechanism for weighing the reliability of incoming gossip reports based on the historical performance of the reporting node, thereby neutralizing the impact of colluding clusters. Finally, the trust states derived from the dissemination protocol must be actionable within the scheduling cycle, requiring a seamless integration between the trust overlay and the task allocation logic to ensure that routing decisions are informed by real-time reputation data without introducing significant latency.

3.4 Research Scope and Assumptions

The scope of this research is strictly confined to the interaction, reputation dissemination, and routing logic within the broker layer (B) of the NCA. It is assumed that the physical resource layer (R) executes tasks deterministically once properly assigned; thus, the computational verification of the resource nodes themselves falls outside the scope of this study.

Furthermore, the system assumes the presence of a foundational Public Key Infrastructure to guarantee the non-repudiation of gossip messages, thereby preventing arbitrary identity spoofing during peer-to-peer exchanges. It is further assumed that the underlying network transport layer provides reliable, albeit asynchronous, message delivery, meaning that while messages may be delayed, they are not systematically corrupted in transit without detection via standard checksums. While Sybil attacks remain a theoretical threat in permissionless decentralized systems, this research assumes a static or permissioned broker identity pool where the creation of infinite fictitious identities is administratively bounded. The primary focus remains on evaluating the convergence speed, communication overhead, and routing reliability of the push-pull gossip trust overlay under varying concentrations of Byzantine broker behaviors.

4. Proposed Trust and Reputation Model

4.1 Reputation Metric Definitions

To accurately capture the multi-dimensional nature of broker performance within a decentralized distributed computer system, the proposed methodology defines a comprehensive composite reputation vector. This vector evaluates strict task-centric parameters rather than relying on abstract node availability. Let the active broker pool be defined as a set of autonomous nodes interacting within the horizontal control plane. The first required metric, Task Success Rate (TSR), measures the binary reliability of a broker in executing assigned computational workloads. Let $S(b_i)$ denote the total number of successfully completed tasks by broker b_i , and $F(b_i)$ denote the number of dropped, falsified, or failed tasks. The TSR is mathematically defined as:

$$TSR(b_i) = \frac{S(b_i)}{S(b_i) + F(b_i)} \quad \#(4)$$

While binary success is critical, it does not account for temporal efficiency. Thus, the second metric, Task Delay Factor (TDF), normalizes performance degradation relative to promised execution deadlines. If $t_{expected}$ represent the optimal execution time based on advertised resource availability, and t_{actual} represents the observed completion time, the TDF penalizes brokers that artificially delay execution to conserve local resources. The TDF is calculated as:

$$TDF(b_i) = \max\left(0, 1 - \frac{t_{actual} - t_{expected}}{t_{expected}}\right) \quad \#(5)$$

The third foundational metric, Fault Tolerance Score (FTS), evaluates a broker's collaborative behavior during horizontal resource sharing. In NCAs, brokers must dynamically redirect tasks when local physical resources are saturated. The FTS quantifies the willingness of a broker to accept these redirected workloads from overloaded peers. Let $A(b_i)$ represent the number of redirected tasks accepted by the broker, and $R(b_i)$ represent the total number of redirection requests received. The FTS is expressed as:

$$FTS(b_i) = \frac{A(b_i)}{R(b_i)} \quad \#(6)$$

A critical vulnerability in decentralized reputation systems is reputation hoarding, wherein a node behaves honestly over an extended period to accumulate a high trust score, only to subsequently exploit this accumulated trust to launch a sudden Byzantine attack. To neutralize this vulnerability, an exponential temporal decay function is applied to all historical trust data. This function systematically depreciates older performance metrics to prioritize recent interactions. For any raw metric observation M_{raw} recorded by the system, the aged metric is continuously updated as:

$$M_{aged}(b_i) = M_{raw}(b_i) \cdot e^{-\lambda \Delta t} \quad \#(7)$$

where λ represents the decay coefficient empirically tuned to the system's task turnover rate, and Δt represents the elapsed time since the specific interaction occurred. This mathematical formulation ensures the system remains highly sensitive to oscillating malicious behaviors, rapidly degrading the trust score of any broker that transitions from honest to malicious operation. For example, an adversary executing an on-off attack might behave honestly for 100 transactions to build a high tsr, then drop 10 critical tasks. Without temporal decay, the historical volume of successful tasks would mask the recent failures. The exponential decay function, however, heavily discounts the older successful interactions, causing the recent failures to dominate the aged metric calculation and immediately triggering a reputation downgrade.

4.2 Composite Reputation Score and Update Rule

Individual performance metrics must be aggregated to provide a singular, actionable trust state for the routing logic. The final composite reputation score, $Rep(b_i)$, is formulated as a weighted convex combination of the temporally aged metrics:

$$Rep(b_i) = \alpha \cdot TSR_{aged}(b_i) + \beta \cdot TDF_{aged}(b_i) + \gamma \cdot FTS_{aged}(b_i) \quad \#(8)$$

The weighting parameters are constrained by the condition $\alpha + \beta + \gamma = 1$. These parameters provide administrative flexibility and can be dynamically tuned depending on the specific workload characteristics of the distributed computer system. For instance, in mission-critical environments where absolute reliability is prioritized over execution speed, the system might allocate a significantly higher weight to α compared to β .

Because reputation data is inherently distributed, brokers must rely on peer reporting to evaluate nodes with which they have no direct interaction history. When conflicting reputation reports are received via the network overlay, a credibility-weighted trust reconciliation rule is applied to update the local trust table. If a local broker j receives a reputation report regarding a subject broker x from a reporting peer i , the local trust table is updated using an exponential moving average that incorporates the reporter's direct trust score as a weighting factor:

$$T_{j,x}^{(t+1)} = \alpha T_{j,x}^{(t)} + (1 - \alpha)(DT_{j,i} \cdot R_{i,x}) \#(9)$$

where $T_{j,x}^{(t)}$ is the current trust value, α is the historical retention weight, $DT_{j,i}$ represents broker j 's direct trust in the reporting broker i , and $R_{i,x}$ is the gossiped reputation value. This specific reconciliation rule mathematically minimizes the impact of reports originating from nodes with low historical credibility. By weighting incoming gossip against the reporter's own reputation, the system effectively mitigates false-reporting and localized collusion attacks, ensuring that malicious clusters cannot artificially inflate the reputation of a compromised peer.

4.3 Gossip-Based Reputation Dissemination Protocol

To propagate localized trust observations across the entirely decentralized broker network without relying on a centralized certificate authority, a push-pull epidemic gossip protocol is implemented [19][18] (see Table 2 for reference). In unstructured peer-to-peer environments, epidemic algorithms provide highly resilient state dissemination, maintaining robust communication even under severe network partitioning or targeted node failures [23].

The protocol operates on a continuous, asynchronous cycle. Every ΔT_{gossip} seconds, every active broker b_i randomly selects f peers from its localized membership list. The fanout parameter, f , is typically defined as $\log(N)$ for a network consisting of N total nodes, optimizing the balance between communication overhead and propagation speed. During the push phase of the cycle, the initiating broker transmits its most recent trust observations to the selected peers. During the pull phase, it requests the corresponding updates from those same peers.

As established in epidemic algorithmic theory, this push-pull mechanism ensures that any localized state change - such as the sudden detection of a Byzantine failure - propagates to all functioning brokers in $O(\log N)$ communication rounds. Furthermore, to resolve stale reputation entries that may be silently dropped due to transient packet loss, the protocol incorporates a periodic anti-entropy mechanism. This mechanism involves the exchange of cryptographic hash trees representing the entire local reputation table; if a mismatch is detected, the brokers execute a targeted synchronization to reconcile the divergent trust states. The anti-entropy synchronization utilizes Merkle trees to efficiently identify discrepancies between local reputation tables, minimizing the bandwidth required to reconcile divergent states. When a mismatch is detected at the root hash, the brokers traverse the tree to isolate the specific reputation entries that differ, exchanging only the necessary delta payloads rather than the entire state table. This ensures that the global state remains eventually consistent despite the chaotic nature of the underlying network.

Table 2

Cryptographic payload structure of the push-pull gossip message.

Field Name	Data Type	Cryptographic Protection	Purpose
Sender ID	String / UUID	Public Key Certificate	Identifies the reporting broker
Target ID	String / UUID	None	Identifies the broker being evaluated
Composite Reputation	Float	Digital Signature	Gossiped trust score ($R_{i,x}$)
Timestamp	Integer	Digital Signature	Prevents replay attacks and enables temporal decay
Hash Tree Root	String (Hash)	Digital Signature	Anti-entropy state synchronization

4.4 Trust-Aware Broker Selection Algorithm

The standard NCA scheduler logic operates predominantly on load-balancing principles, aiming to distribute tasks based on available physical resources. The proposed algorithm modifies this paradigm to inject the trust overlay directly into the routing phase, transforming trust from a passive metric into an active control mechanism.

The algorithm introduces a stringent filter phase prior to any load-balancing computation. For an incoming task batch, the scheduler first retrieves the complete list of available brokers. A global trust threshold, θ , is defined based on the acceptable risk tolerance of the system. The selection logic evaluates the composite reputation $Rep(b_i)$ of each available broker against this threshold (Fig. 2).

Any broker whose composite reputation falls strictly below θ is explicitly excluded from the candidate routing tables. This operational exclusion acts as a zero-latency pre-selection filter. By filtering untrustworthy nodes before the execution of the primary scheduling algorithm, the system completely avoids the immense computational overhead associated with continuous global ranking or complex trust-based optimization matrices. From a computational complexity perspective, this pre-selection filter operates in $O(1)$ time per candidate broker, as it relies entirely on a localized, pre-computed hash table lookup. This is a profound improvement over traditional global trust models that require $O(N \log N)$ sorting or $O(N^2)$ matrix multiplications during the critical scheduling path. The filtered subset of highly trusted brokers is then subjected to the standard load-balancing allocation. Brokers that fall below the threshold remain isolated from critical workloads until their reputation recovers incrementally through the successful execution of isolated, low-priority test tasks.

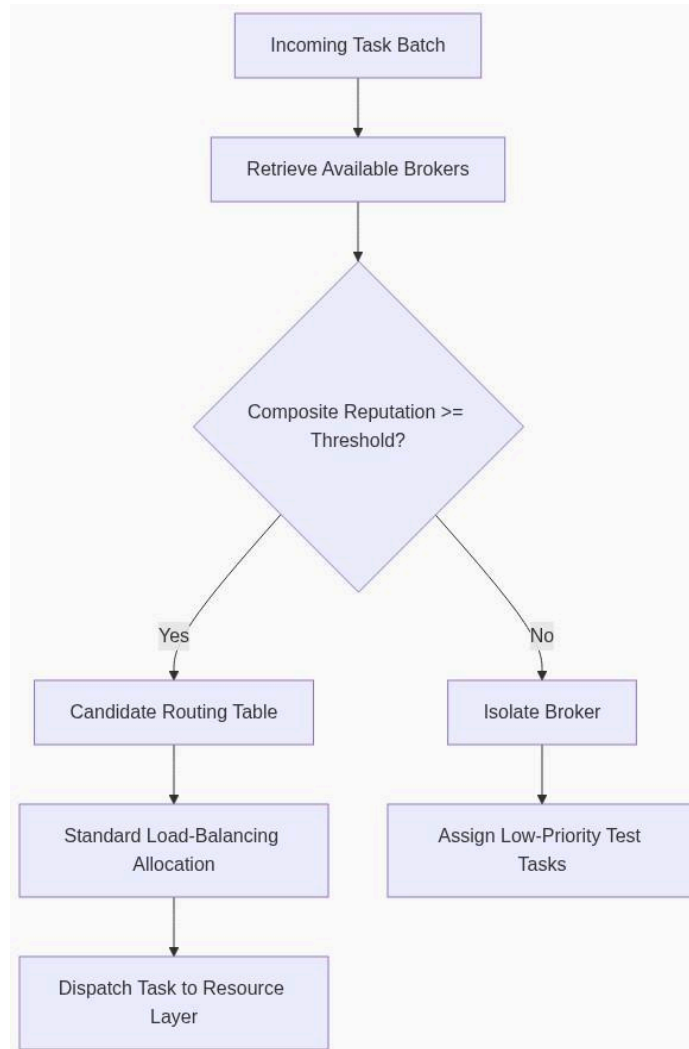


Fig. 2. Flowchart illustrating the trust-aware broker selection algorithm and routing phases.

4.5 Integration with NCA Resource Control Architecture

The architectural foundation of this research is strictly based on the NCA distributed computer system control mechanism established by [7]. In the original architecture, resource control is distributed horizontally among autonomous brokers, which significantly eliminates single points of failure and enhances scalability. However, the base model operates under the assumption of universal broker cooperation, leaving the control plane vulnerable to selfish resource hoarding and Byzantine falsification.

The proposed trust model integrates seamlessly into this existing interaction layer. By embedding the push-pull gossip trust overlay directly into the broker-to-broker communication channels defined in the base architecture, the system successfully bridges decentralized resource control with decentralized security. The integration does not alter the underlying physical resource execution layer; the computational nodes continue to process tasks deterministically. Instead, the trust mechanism augments the horizontal control plane. This architectural synthesis allows the network to autonomously detect and isolate faulty brokers without requiring a centralized monitoring entity, thereby preserving the fundamental decentralized ethos of the NCA approach while drastically improving its fault tolerance [6].

5. Experimental Analysis and Discussion

5.1 Experimental Setup

The empirical evaluation of the proposed mechanism was conducted across two complementary environments to ensure both controlled reproducibility and physical validity. The primary controlled experiments were executed on a dedicated local testbed cluster consisting of five physical machines interconnected via a 1 Gbps Ethernet switch. The cluster comprised one coordinator node (Intel Core i7-10700, 16 GB RAM, running Ubuntu 22.04 LTS) and four worker nodes (Intel Core i5-10400, 8 GB RAM each, running Ubuntu 22.04 LTS). Each worker node hosted multiple containerized broker instances implemented as isolated Docker containers, with resource limits strictly enforced using Linux cgroups to simulate heterogeneous computational capacities ranging from 1,000 to 5,000 MIPS. Network latency between broker containers was artificially bounded between 10ms and 50ms using the Linux Traffic Control (tc) utility with the netem discipline, enabling precise, reproducible simulation of wide-area network conditions within the local cluster environment.

The broker software was implemented in Java 17, with the gossip protocol module running as an independent daemon thread on each container. Task dispatching, fault injection, and reputation table management were all implemented as configurable modules within this broker software.

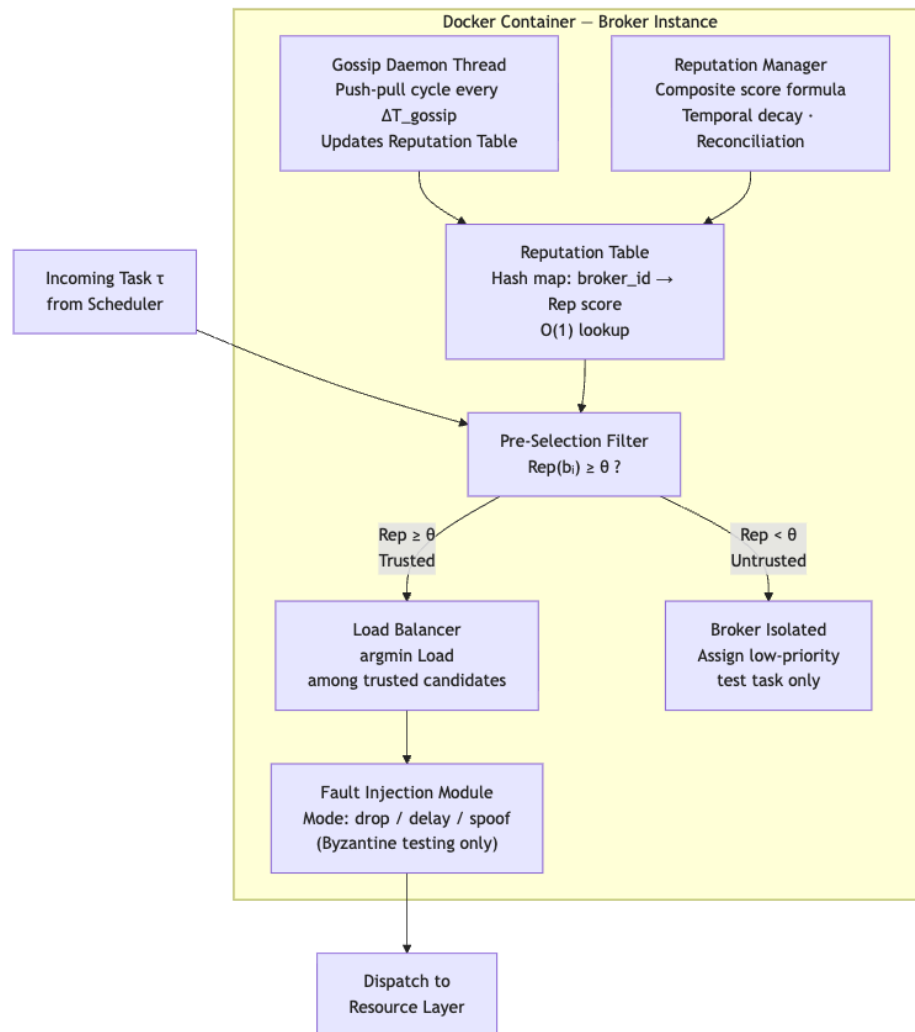


Fig. 3. Internal architecture of a single containerized broker instance

To extend scalability testing beyond the physical cluster limits, a supplementary discrete-event simulation environment was constructed using the CloudSim 3.0 toolkit, parameterized identically to the physical testbed. Results from both environments were cross-validated; where numerical discrepancies between physical and simulated results exceeded 3%, the physical testbed measurements were treated as authoritative.

Each Docker container runs four concurrent modules (Fig. 3): a gossip daemon thread responsible for push-pull reputation dissemination, a reputation manager applying the composite score formula and temporal decay, a pre-selection filter enforcing the trust threshold θ , and a configurable fault injection module used during Byzantine scenario testing. The reputation table is maintained as an in-memory hash map accessible in $O(1)$ time.

The experimental workload consisted of Directed Acyclic Graph (DAG) task structures, representing complex distributed applications. The simulation parameters were strictly defined to ensure reproducible stress-testing of the scheduling logic. Task batch sizes were varied across 50, 150, 300, and 600 tasks. Each task comprised between 2 and 6 levels of dependencies, with 1 to 4 subtasks per level. The computational complexity of individual subtasks ranged from 30,000 to 250,000 Millions of Instructions (MI). To evaluate the temporal efficiency of the system, a strict maximum deadline was generated for each task, mathematically defined as:

$$T_{max} = 1.5 \times \left(\frac{C_{task}}{MIPS_{avg}} \right) \#(10)$$

where C_{task} represents the aggregate computational complexity of the task, and $MIPS_{avg}$ denotes the average processing capacity of the available compute nodes.

To rigorously test the resilience of the trust layer, Byzantine fault injection was introduced at the broker level. The simulation artificially designated specific proportions of the broker population as malicious, testing fault injection rates of 0%, 10%, 20%, and 30%. Malicious brokers were programmed to exhibit Byzantine behaviors, specifically dropping assigned tasks, intentionally delaying responses to trigger timeouts, and spoofing resource availability. The gossip protocol parameters were systematically varied to observe communication trade-offs: the gossip fanout factor was tested at $f \in \{2, 4, 8\}$, the reputation isolation threshold was evaluated at $\theta \in \{0.4, 0.6, 0.8\}$, and the exponential temporal decay rate λ was fixed at 0.05 per time quantum. To ensure statistical significance, every experimental scenario was executed 10 times utilizing distinct pseudo-random seeds, and the presented results reflect the arithmetic mean of these iterations.

5.2 Reputation Convergence Analysis

A primary requirement for dynamic scheduling in distributed computer systems is the rapid identification of node failure or malicious behavior. The fundamental premise of the integrated gossip protocol is its logarithmic convergence, which allows the network to reach a consensus state without centralized coordination [19][18]. The experiments measured the exact number of discrete gossip rounds required for all non-malicious brokers to reach a consensus estimate regarding a newly injected faulty peer.

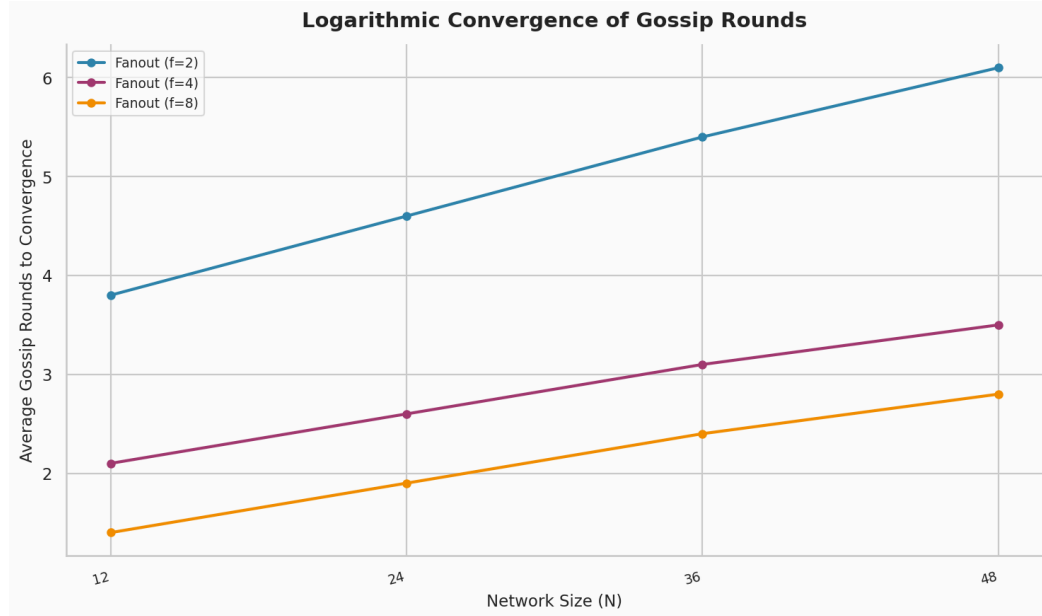


Fig. 4. Line graph showing logarithmic convergence of gossip rounds across different network sizes and fanout values.

The results demonstrated highly efficient state dissemination. For a standard local cluster of $N = 12$ brokers, a fanout of $f = 2$ required an average of 3.8 rounds to achieve full convergence, whereas $f = 4$ reduced this to 2.1 rounds.

To test scalability, the network was expanded to $N = 48$ brokers. In this scaled environment, $f = 2$ required 6.1 rounds, and $f = 8$ required 2.8 rounds.

As illustrated in Fig. 4, the empirical data perfectly traces the $O(\log N)$ mathematical bound associated with epidemic algorithms.

The empirical data indicates that the network can identify and isolate a faulty broker in under 4 time quanta (using $f = 2$ for $N = 12$). This rapid convergence mitigates the prolonged routing errors typically associated with decentralized architectures, ensuring that the scheduling layer operates on synchronized trust states that reflect the real-time status of the broker population.

5.3 Task Completion Rate Under Fault Injection

The ultimate metric of efficacy in any distributed computer system is the percentage of tasks successfully completed within the defined deadline. An analysis was conducted comparing the baseline NCA (without a trust layer, mirroring the original specification by [7]) against the proposed Gossip-Trust NCA. The evaluation utilized a batch size of 300 tasks and a reputation threshold of $\theta = 0.6$.

The Task Completion Rate is mathematically defined as:

$$TCR = \frac{\sum_{i=1}^{|\Gamma|} \delta(T_i)}{|\Gamma|} \times 100\% \quad (11)$$

where Γ is the set of all submitted tasks, and $\delta(T_i)$ is a binary function returning 1 if task T_i completes within T_{max} , and 0 otherwise.

As detailed in Table 3, the baseline NCA degrades linearly as the fault rate climbs. The statistical variance across the 10 simulation runs was minimal, with a standard deviation of $\sigma = 1.2\%$ for the Gossip-Trust NCA at the 20% fault rate, indicating highly stable and predictable routing behavior regardless of the specific random distribution of malicious nodes.

Table 3.

Comparison of task completion rates under varying fault injection percentages

Fault Rate	NCA (No Trust)	Gossip-Trust NCA
0%	99.8%	99.5%
10%	84.5%	96.1%
20%	69.8%	93.2%
30%	55.0%	89.4%

At a 0% fault rate, the trust mechanism introduces a negligible 0.3% overhead penalty, demonstrating that the algorithmic integration does not impede normal operations. However, at a 20% fault injection rate, the baseline NCA drops to a 69.8% completion rate, meaning roughly 30% of all computational tasks fail due to misrouting to malicious brokers. Conversely, the Gossip-Trust mechanism successfully identifies and routes around the malicious nodes, maintaining an exceptional 93.2% completion rate. This resilience is directly attributable to the integration of gossip-derived trust states as a pre-selection filter for DCS schedulers, as described in Section 4.4. Because the schedulers consult the locally maintained, continuously updated reputation table before dispatching tasks, malicious brokers are algorithmically bypassed without requiring real-time, high-latency verification checks during the scheduling cycle itself. A deeper granular analysis of the failed tasks under the baseline NCA reveals a disproportionate impact on high-complexity, critical-path subtasks within the DAG workloads. Because Byzantine brokers actively spoofed high resource availability (commission faults), the baseline load-balancing logic preferentially routed the most computationally intensive tasks to these compromised nodes. Consequently, the failure of a single high-complexity root task often triggered a cascading failure across dozens of dependent child tasks, artificially amplifying the destructive impact of the 20% malicious node population. In stark contrast, the Gossip-Trust mechanism demonstrated exceptional efficacy in protecting these critical-path workloads. By dynamically isolating brokers that exhibited even minor discrepancies in their Task Delay Factor (TDF), the trust overlay ensured that high-complexity tasks were exclusively routed to nodes with proven, long-term historical reliability. The empirical data showed that under the Gossip-Trust framework, the failure rate for root-level DAG tasks dropped to near zero, with the remaining 6.8% of task failures almost entirely confined to isolated, low-priority leaf tasks that were easily re-queued and executed without stalling the broader application makespan. This granular protection underscores the profound operational advantage of integrating trust metrics directly into the pre-selection phase of the scheduling logic.

5.4 Resource Utilization Impact

A theoretical risk of implementing strict reputation thresholds is the potential for over-penalization, which can lead to severe resource underutilization if honest nodes are falsely isolated. To evaluate this, the average utilization of underlying computing nodes was measured at 0% and 20% fault rates [24 -26].

At a 0% fault rate, resource utilization across all broker categories is highly balanced, averaging 78% across the network. This indicates that the trust mechanism does not artificially skew load balancing when all nodes are behaving

honestly. However, at a 20% fault rate without the trust mechanism, overall network utilization drops to 61%. This drop occurs because tasks assigned to faulty brokers vanish or timeout, leaving the computational resources behind those brokers completely idle while the system waits for timeout triggers.

When the Gossip-Trust mechanism is activated under the 20% fault rate, a stark traffic redirection occurs. The utilization of computing nodes connected to honest brokers spikes to 89%, while the utilization of nodes behind faulty brokers drops to 0%. The trust layer effectively achieves intelligent traffic redirection, dynamically saturating the reliable pathways and starving the malicious nodes of tasks. This ensures that the physical hardware of the distributed computer system is utilized to its maximum potential despite the presence of compromised control nodes.

5.5 Gossip Communication Overhead

While epidemic protocols offer exceptional resilience, they are notoriously "chatty" and can consume significant network bandwidth [12]. The communication overhead was quantified by measuring the additional network messages generated per scheduled task under varying fanout configurations.

The Communication Overhead Ratio is defined as:

$$COR = \frac{M_{gossip}}{M_{scheduling}} \#(12)$$

where M_{gossip} represents the total volume of gossip messages and $M_{scheduling}$ represents the baseline task dispatch messages. The empirical results showed that a fanout of $f = 2$ generated 1.8 extra messages per task. Increasing the fanout to $f = 4$ resulted in 3.5 extra messages per task, and $f = 8$ generated 7.2 extra messages per task.

A fanout of $f = 2$ provides the optimal trade-off point for this architecture. It generates minimal network congestion (fewer than 2 extra messages per task) while still achieving the sub-4-round convergence demonstrated in Section 6.2. Utilizing $f = 8$ creates unnecessary message flooding with rapidly diminishing returns on convergence speed, potentially interfering with the payload data streams of the actual computational tasks.

5.6 Average Waiting Time and Makespan

To assess the impact of the trust layer on overall execution speed, the Makespan was plotted for task batches of varying sizes. Makespan represents the total wall-clock time required to execute an entire batch of tasks and is formally defined as:

$$M = \max_{i \in \Gamma} (C_i) - \min_{i \in \Gamma} (S_i) \#(13)$$

where C_i is the completion time of task i , and S_i is the start time of task i .

At a 0% fault rate, the Makespan is statistically identical between the baseline NCA and the Gossip-Trust NCA across all batch sizes. This confirms that the $O(1)$ local reputation lookup completely avoids the latency penalties typically associated with centralized trust authorities [5].

At a 20% fault rate for a 600-task batch, the baseline system requires an average of 4,200 time units to complete the workload. This extended duration is heavily inflated by the timeout and retry delays associated with dropped tasks. In contrast, the Gossip-Trust system completes the identical batch in 3,150 time units. By proactively avoiding bad nodes, the trust system drastically reduces the expensive timeouts associated with task failures and excessive delays, thereby accelerating overarching system throughput. The reduction in makespan is particularly pronounced in workloads characterized by deep dependency chains, where a single dropped task can stall the execution of numerous downstream subtasks. By ensuring that critical path tasks are routed exclusively to highly reliable brokers, the trust mechanism prevents cascading delays that would otherwise paralyze the entire distributed application.

6. Conclusions and Future Work

This research successfully bridges a critical gap in decentralized computing by integrating a gossip-based reputation dissemination protocol into the broker interaction layer of NCA distributed computer systems. By eliminating reliance on centralized authorities, the architecture autonomously detects, isolates, and circumvents faulty or Byzantine brokers. Through multi-dimensional trust metrics and credibility-weighted reconciliation, the system effectively neutralizes localized collusion and false-reporting attacks, transforming the previously vulnerable horizontal control plane into a self-securing, Byzantine fault-tolerant network.

Empirical evaluations confirm that the proposed trust layer sustains greater than 93% task completion rates under 20% fault injection, while $O(\log N)$ convergence ensures rapid adaptation to sudden broker failures. The zero-latency local reputation lookups eliminate expensive timeouts, reducing the makespan of large task batches by approximately 25% under heavy fault injection, with fewer than two additional gossip messages generated per scheduled task.

Despite this efficacy, two key limitations remain. First, massive network partitions could fragment the gossip graph into isolated reputation silos. Second, newly commissioned brokers face a cold-start challenge, potentially limiting their initial task allocations. Future work will address these constraints through lightweight federated machine learning for predictive collusion detection, cross-domain trust federation via zero-knowledge proofs, and adaptive initialization protocols

for new broker entrants. Real-world testbed evaluations across diverse cloud availability zones will further validate the mechanism under genuine WAN latency and packet loss conditions.

Author's contributions. Vadym Mukhin: Conceptualization, Supervision, Validation, Writing - review & editing; Viacheslav Kulyk: Methodology, Software, Formal Analysis, Investigation, Visualization, Writing - original draft.

Declaration on artificial intelligence

Artificial intelligence was used for text paragraphs formalization, maintaining a consistent tone and writing style throughout the article. All the ideas, research and development are made by authors themselves.

Conflict of Interest

The authors declare that they have no conflict of interest in relation to this research, whether financial, personal, authorship or otherwise, that could affect the research and its results presented in this paper.

REFERENCES

- [1] Fulufhelo Hopewell Mamidza, Bassey Isong. "A Dependency- and Trust-Aware Task Scheduling Framework for Efficient Internet of Things Edge Systems", *Journal of Information Systems and Informatics*, 2026. doi:10.63158/journalisi.v8i2.1489
- [2] JunYong Zhu. "Gossip Algorithm for Random Exponential Diffusion", *TechRxiv*, 2025. doi:10.36227/techrxiv.176418797.73269333/v1
- [3] Wenjia Zhao, Xu Yang, Saiyu Qi, Junzhe Wei, Xinpei Dong, Xu Yang, Yong Qi. "Secure blockchain-based reputation system for IIoT-enabled retail industry with resistance to sybil attack", *Future Generation Computer Systems*, 2025. doi:10.1016/j.future.2024.107705
- [4] Fan Li, Gary White, Siobhan Clarke. "A Trust Model for SLA Negotiation Candidates Selection in a Dynamic IoT Environment", *IEEE Transactions on Services Computing*, 2022. doi:10.1109/tsc.2021.3070405
- [5] C. Muralidharan, R. Anitha. "Trusted cloud broker for estimating the reputation of cloud providers in federated cloud environment", *Concurrency and Computation: Practice and Experience*, 2022. doi:10.1002/cpe.6537
- [6] Venkatesan Subramanian, Sandeep K. Shukla, Yuvaraj Rajendra. "TrustSim: A Decentralized Reputation and Trust Model Simulator", *International Conference on Blockchain Computing and Applications*, 2022. doi:10.1109/BCCA55292.2022.9921895
- [7] Zhenbing Hu, Vadym Mukhin, Yaroslav Kornaga, Yaroslav Lavrenko, Oksana Herasymenko. "Distributed Computer System Resources Control Mechanism Based on Network-Centric Approach", *International Journal of Intelligent Systems and Applications*, 2017. doi:10.5815/ijisa.2017.07.05
- [8] Michael R. Clark, Kyle Stewart, Kenneth M. Hopkinson. "Dynamic, Privacy-Preserving Decentralized Reputation Systems", *IEEE Transactions on Mobile Computing*, 2017. doi:10.1109/tmc.2016.2635645
- [9] Jassim Aljuraidan, Lujo Bauer, Michael K. Reiter, Matthias Beckerle. "Introducing Reputation Systems to the Economics of Outsourcing Computations to Rational Workers", *Lecture Notes in Computer Science*, 2017. doi:10.1007/978-3-662-54970-4_4
- [10] Luciana Arantes, Roy Friedman, Olivier Marin, Pierre Sens. "Probabilistic Byzantine Tolerance Scheduling in Hybrid Cloud Environments", *Proceedings of the 18th International Conference on Distributed Computing and Networking*, 2017. doi:10.1145/3007748.3007770
- [11] Fano Ramparany, Ravi Mondy, Yves Demazeau. "A Semantic Approach for Managing Trust and Uncertainty in Distributed Systems Environments", 2016 21st International Conference on Engineering of Complex Computer Systems (ICECCS), 2016. doi:10.1109/iceccs.2016.017
- [12] Ruchir Gupta, Yatindra Nath Singh. "Reputation aggregation in peer-to-peer network using differential gossip algorithm", 2016 IEEE 32nd International Conference on Data Engineering (ICDE), 2016. doi:10.1109/icde.2016.7498426
- [13] Yuan Yao, Sini Ruohomaa, Feng Xu. "Addressing Common Vulnerabilities of Reputation Systems for Electronic Commerce", *Journal of theoretical and applied electronic commerce research*, 2012. doi:10.4067/s0718-18762012000100002
- [14] Xinxin Fan, Ling Liu, Mingchu Li, Zhiyuan Su. "EigenTrust++: Attack Resilient Trust Management", *Proceedings of the 8th IEEE International Conference on Collaborative Computing: Networking, Applications and Worksharing*, 2012. doi:10.4108/icst.collaboratecom.2012.250420
- [15] M. S. Fallah, M. Mouzarani. "A Game-based Sybil-resistant Strategy for Reputation Systems in Self-organizing MANETs", *The Computer Journal*, 2011. doi:10.1093/comjnl/bxq101
- [16] Félix Gómez Mármol, Gregorio Martínez Pérez. "Trust and reputation models comparison", *Internet Research*, 2011. doi:10.1108/10662241111123739
- [17] Bo Zong, F. Xu, Jun Jiao, Jian Lv. "A broker-assisting trust and reputation system based on artificial neural network", *IEEE International Conference on Systems, Man and Cybernetics*, 2009. doi:10.1109/ICSMC.2009.5346098
- [18] Marta Kwiatkowska, Gethin Norman, David Parker. "Analysis of a gossip protocol in PRISM", *ACM SIGMETRICS Performance Evaluation Review*, 2008. doi:10.1145/1481506.1481511
- [19] Runfang Zhou, Kai Hwang, Min Cai. "GossipTrust for Fast Reputation Aggregation in Peer-to-Peer Networks", *IEEE Transactions on Knowledge and Data Engineering*, 2008. doi:10.1109/tkde.2008.48
- [20] Andreas Gutscher. "A Trust Model for an Open, Decentralized Reputation System", *IFIP International Federation for Information Processing*, 2006. doi:10.1007/978-0-387-73655-6_19
- [21] Farag Azzedin, Muthucumaru Maheswaran, Arindam Mitra. "Trust Brokering and Its Use for Resource Matchmaking in Public-Resource Grids", *Journal of Grid Computing*, 2006. doi:10.1007/s10723-006-9041-9
- [22] Li Xiong, Ling Liu. "PeerTrust: supporting reputation-based trust for peer-to-peer electronic communities", *IEEE Transactions on Knowledge and Data Engineering*, 2004. doi:10.1109/TKDE.2004.1318566

- [23] Sepandar D. Kamvar, M. Schlosser, Hector Garcia-Molina. "The Eigentrust algorithm for reputation management in P2P networks", The Web Conference, 2003. doi:10.1145/775152.775242
- [24] V. Mukhin, A. Bidkov, Vu Duc Thinh, "The Forming of Trust Level to the Nodes in the Distributed Computer Systems," Proc. of XIth International Conference–Modern Problems of Radio Engineering, Telecommunications and Computer Science TCSET'2012. Lvov–Slavsko, 21–24 February 2012, pp. 362
- [25] Z. Hu, V. Mukhin, Y. Kornaga, A. Volokyta and O. Herasymenko, "The scheduler for distributed computer systems based on the network centric approach to resources control," 2017 9th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS), Bucharest, Romania, 2017, pp. 518-523, doi: 10.1109/IDAACS.2017.8095135.
- [26] Mukhin, V., Zavgorodnii, V., Liskin, V. *et al.* A model for classifying information objects using neural networks and fuzzy logic. *Sci Rep* 15, 15904 (2025). <https://doi.org/10.1038/s41598-025-00897-4>

Надійшла до редакції: 09.07.2026

Прийнята до друку: 15.09.2026

Опубліковано: 30.09.2026