Computer Networks 282 (2026) 112242

Contents lists available at ScienceDirect

Computer Networks

journal homepage: www.elsevier.com/locate/comnet

Review article

RepuNet: A Reputation System for Mitigating Malicious Clients in DFL

Isaac Marroquí Penalva, Enrique Tomás Martínez Beltrán *, Manuel Gil Pérez, Alberto Huertas Celdrán

Department of Information and Communications Engineering, University of Murcia, Murcia, 30100, Spain

ARTICLE INFO

Keywords:
Federated learning
Reputation system
Decentralized learning
Malicious clients
Model poisoning
Attack mitigation

ABSTRACT

Decentralized Federated Learning (DFL) enables nodes to collaboratively train models without a central server, introducing new vulnerabilities since each node independently selects peers for model aggregation. Malicious nodes may exploit this autonomy by sending corrupted models (model poisoning), delaying model submissions (delay attack), or flooding the network with excessive messages, negatively affecting system performance. Existing solutions often rely on rigid configurations or additional infrastructure, such as blockchains, which can incur computational overhead, introduce scalability issues, or limit adaptability. To overcome these limitations, this paper proposes RepuNet, a decentralized reputation system that categorizes threats in DFL and dynamically evaluates node behavior using metrics like model similarity, parameter changes, message latency, and communication volume. Node influence in model aggregation is adjusted based on each node's reputation score. RepuNet was integrated into the Nebula platform and experimentally evaluated with MNIST and CIFAR-10 datasets under non-IID distributions, using federations of up to 25 nodes in both fully connected and random topologies. The evaluation considers different attack intensities, frequencies, and activation intervals, and includes comparisons with Byzantine-resilient aggregation mechanisms (Krum and Trimmed Mean), stronger structured poisoning strategies (Signed Neuron Remapping and GLL Neuro Inversion), as well as an ablation study of the exclusion threshold and a communication overhead analysis. Results demonstrate that RepuNet effectively detects and mitigates malicious behavior, achieving F1 scores above 95% on MNIST and approximately 76% on CIFAR-10. These outcomes highlight the adaptability, robustness, and practical potential of RepuNet for mitigating threats in decentralized environments.

1. Introduction

Federated Learning (FL) has emerged as a key paradigm for collaborative model training without sharing raw data, addressing growing concerns over data privacy in the era of Artificial Intelligence (AI) [1]. Traditionally, centralized FL architectures have dominated this field, but their dependence on a single coordinator introduces critical vulnerabilities, as the central server can become a bottleneck or a target for adversarial manipulation. To mitigate these risks, Decentralized Federated Learning (DFL) has gained increasing attention by removing the central authority, distributing coordination across peers, and enhancing resilience against single-point failures. For instance, in healthcare collaborations in which hospitals jointly train diagnostic models without exchanging patient data, a delayed or poisoned update from a single participant can significantly degrade model reliability and compromise clinical outcomes.

However, the absence of centralized supervision in DFL introduces distinct security challenges. In the first phase, nodes independently train local models while being exposed to model poisoning attacks. In the communication phase, nodes exchange parameters with their neighbors, making them vulnerable to delay attacks. Finally, during aggregation, malicious nodes may execute flooding or denial-of-service attacks. These coordinated threats highlight the need for dynamic, adaptive mechanisms to ensure trust and robustness in decentralized training environments.

Existing defenses against such attacks include blockchain-based trust management, cluster-based aggregation, suspicious model filtering, and cryptographic techniques. Nevertheless, these solutions face significant drawbacks: blockchain-based approaches incur high computational and storage overhead; cluster-based approaches require careful configuration and are susceptible to collusion; and cryptographic or filtering schemes can reduce scalability or increase false positives. Despite these advances, no existing approach provides a lightweight, fully decentralized mechanism that can adaptively evaluate trust under dynamic DFL conditions without incurring substantial overhead. This limitation defines the research gap motivating the design of **RepuNet**, a threat-

* Corresponding author.
E-mail addresses: i.marroquipenalva@um.es (I. Marroquí Penalva), enriquetomas@um.es (E.T. Martínez Beltrán), mgilperez@um.es (M. Gil Pérez), alberto.huertas@um.es (A. Huertas Celdrán).

https://doi.org/10.1016/j.comnet.2026.112242
Received 18 November 2025; Received in revised form 3 March 2026; Accepted 20 March 2026
Available online 27 March 2026

aware, adaptive reputation system tailored for decentralized environments. Unlike blockchain-based or Byzantine aggregation schemes, RepuNet manages reputation and trust without requiring consensus layers or fixed thresholds, maintaining flexibility and low computational cost. It continuously monitors node behavior across rounds to identify communication delays, abnormal update patterns, and excessive traffic, penalizing untrustworthy participants while maintaining training stability.

The main contributions of this work are as follows:

- **Design of a threat-aware and adaptive reputation system for DFL.** RepuNet integrates multiple behavioral metrics, such as model similarity, parameter variation, latency, and message volume, to dynamically evaluate and weight node reliability, enabling early detection of malicious behavior.
- **Progressive penalization and controlled exclusion mechanism.** Nodes with consistently low reputation have reduced aggregation influence but are not permanently excluded, supporting reintegration once behavior stabilizes. This controlled exclusion allows continuous participation without jeopardizing global convergence.
- **Comprehensive evaluation on the Nebula platform under realistic DFL settings.** RepuNet is implemented and validated on the Nebula platform using configurable topologies, attack scenarios, and real-time metric visualization. Experiments on fully connected and random topologies, under model poisoning, delay, and flooding attacks, demonstrate up to a 35% higher attack-detection accuracy and a stable global accuracy above 97% under non-IID conditions.

The remainder of this paper is organized as follows. Section 2 reviews related work on DFL security. Section 3 introduces the threat model and attack scenarios. Section 4 details the RepuNet framework and workflow. Section 5 describes the experimental setup and performance metrics. Section 6 presents and discusses the results. Finally, Section 7 concludes the paper and outlines future work.

2. Related work

This section reviews reputation systems in FL and defense mechanisms against malicious behavior in decentralized networks. It covers blockchain-based FL, hierarchical aggregation, and attack-detection strategies, including data and model poisoning and DDoS, as well as both centralized and decentralized approaches that promote honest participation and penalize adversarial nodes.

2.1. Reputation systems in federated learning

A centralized approach [2] updates reputation scores based on behavioral history using a beta distribution. Nodes must exceed an initial reputation threshold to participate, ensuring reliability from the outset. Decentralized systems offer transparency and resilience. Kang et al. [3] stores contributions on the blockchain; reputation uses direct and indirect feedback. Zhao et al. [4] applies blockchain reputation in IoT-FL to promote cooperation. Cooperative incentives have emerged. Domingo-Ferrer et al. [5] proposes a co-utility rule system to reward compliant behavior without central coordination. Panigrahi et al. [6] explores hierarchical aggregation using accuracy and participation to select models. [7] proposes FGFL, a blockchain system using reputation and contribution metrics to adjust incentives. Together, these approaches reflect a shift from static, centralized schemes toward adaptive, incentive-aware, and decentralized reputation frameworks suitable for heterogeneous federated learning environments.

2.2. Attack mitigation strategies

DFL is vulnerable to various attacks that compromise the integrity, efficiency, or convergence of the learning process. These include model manipulation, flooding, and message delays. Several studies address model poisoning and data manipulation. Nguyen et al. [8] studies poisoning in intrusion detection and proposes model filtering. In smart grid contexts, [9] employs deep learning to detect falsified data without compromising privacy. To mitigate network saturation, [10] proposes a decentralized FL-based architecture for anomaly detection in industrial IoT. Likewise, [11] demonstrates how distributed traffic analysis in IoT-Edge environments can identify malicious patterns while preserving data locality. Delay attacks, in which nodes transmit outdated updates to desynchronize the network, are addressed in [12], which uses hierarchical FL to mitigate the influence of slower participants. Hallaji et al. [13] surveys anomaly detection, cryptography, and reputation mechanisms. Blockchain has also been proposed to enhance trust and traceability. For example, Yuan et al. [14] uses sharding and layered validation to secure aggregation, while Wang et al. [15] targets passive threats such as lazybone nodes through a multidimensional evaluation of contribution quality. These strategies highlight the need for hybrid defenses that combine anomaly detection, contribution filtering, decentralized coordination, and adaptive reputation to ensure resilience across diverse DFL scenarios.

2.3. Comparison with existing approaches

RepuNet enables decentralized reputation, where nodes assess neighbors without a central authority or a blockchain. Reputation evolves dynamically from local metrics, including model similarity, parameter variation, latency, and message volume, using a neutral initial value to avoid bias. Unlike static or irreversible schemes, RepuNet allows nodes to recover reputation through consistent behavior, balancing penalization and reintegration. Compared to blockchain-based systems, RepuNet avoids the heavy computational and communication overhead. It eliminates the need for immutable global records and relies on lightweight, locally applied updates at each node. Additionally, peer feedback can be optionally incorporated to strengthen individual assessments, especially in scenarios with partial observations, newly joined nodes, or sparse topologies. This combination enhances system resilience without compromising decentralization. As shown in Table 1, RepuNet integrates more metrics, including communication anomalies, and avoids overhead from blockchain-based systems.

3. Threat model

DFL faces several vulnerabilities due to the autonomy of participating nodes, which may act maliciously and compromise the quality, efficiency, or stability of the collaborative process.

The most relevant threats in DFL can be grouped into four categories:

- **Model-based attacks:** manipulation of local parameters (e.g., *model poisoning*, *backdoor attacks*).
- **Communication-based attacks:** such as *delay attacks*, *flooding*, or DoS.
- **Data-based attacks:** noise injection, false data submission, or inference of private information.
- **Identity/participation attacks:** including *Sybil*, *free-riding*, or *lazybone* behaviors.

This work targets three threats due to their impact and the feasibility of local detection:

- **Model poisoning:** injection of malicious updates to degrade the global model.
- **Delay attack:** intentional delay in sending models to disrupt synchronization.
- **Flooding attack:** excessive message generation to saturate the network.

Table 1
Comparison of anomaly detection criteria, detected attacks, and mitigation strategies in FL systems.

Work (Year)	FL Type	Detection / Penalization Criteria	Feedback	Detected Attack(s)	Anomaly Detection	Action Taken
[4] (2019)	DFL (Blockchain)	Crowdsourced contribution history on blockchain	X	–	X	Trustworthy nodes rewarded; reputation affects participation
[3] (2019)	DFL (Blockchain)	Trust via contribution quality and peer reputation (direct/indirect)	✓	–	X	Adjusts reputation; low scores discourage future inclusion
[8] (2019)	IoT	Model deviation from expected behavior (filtering)	X	Poisoning	✓	Suspicious updates discarded before aggregation
[5] (2021)	DFL (Blockchain)	Violation of cooperation rules (coutility model)	X	–	X	Rewards or penalties applied via rule compliance
[2] (2022)	CFL	Deviation in behavioral history (beta-based reputation)	X	–	X	Reputation score adjusted (centralized); no exclusion
[7] (2022)	DFL (Blockchain)	Quality and frequency of valid contributions	X	Lazy / low-effort participation	✓	Contributors receive dynamic rewards or are excluded if passive
[9] (2022)	Smart Grid	Anomalous data pattern detection using deep learning	X	FDIA (false data injection)	✓	Classifier-based filtering of malicious updates
[10] (2022)	Industrial IoT	Traffic anomalies detected via federated classifiers	X	DDoS	✓	Detection used to alert and reconfigure network paths
[11] (2023)	IoT Edge	Distributed traffic anomaly pattern detection in IoT	X	DDoS, poisoning	✓	Local anomaly response; contributes to distributed decision-making
[6] (2023)	Hierarchical CFL	Accuracy, consistency, and past participation history	X	–	X	Contribution filtered before aggregation based on reputation
[13] (2024)	DFL (Blockchain)	Overview of multiple metrics: gradients, consistency, etc. (survey)	X	Poisoning, backdoor, Sybil, free-riding	✓	Describes possible responses: filtering, exclusion, weight adjustment
[12] (2024)	FL (Hierarchical)	Model submission latency (ranking-based classification)	X	Delay	X	Low-ranked nodes deprioritized or excluded in hierarchy
[14] (2025)	DFL (Blockchain)	Distributed model validation through consensus (sharded blockchain)	X	Sybil, poisoning	✓	Invalid models rejected during consensus verification
[15] (2025)	CFL	Contribution scoring via loss, gradient norm, and activity level	✓	Lazybone	✓	Low-contributing nodes excluded from aggregation
RepuNet (2025)	DFL	Similarity, parameter change, arrival latency, message volume	✓	Poisoning, delay, flooding	✓	Dynamic penalization of reputation, exclusion from aggregation, and adaptive reintegration

In the context of model poisoning, we consider both simple perturbation strategies and stronger structured manipulation techniques. In particular, we evaluate two structured variants: ***Signed Neuron Remapping (SNR)***, where neurons are permuted toward representation-opposed counterparts with sign inversion while preserving weight norms, and ***GLL Neuro Inversion***, which performs structured inversion of neuron groups to disrupt gradient alignment without introducing large-magnitude anomalies. These strategies preserve certain statistical properties of the model while inducing structural misalignment, challenging magnitude-based and distance-based aggregation defenses.

These attacks degrade model quality, synchronization, or communication, and can be detected using local metrics such as model similarity, latency, or message count. They are common in real-world deployments. Other threats (e.g., inference, Sybil, or passive participation) require identity validation or secure exchanges and are beyond the scope of this work. By focusing on locally observable threats, the proposed system remains practical and suitable for controlled experimental validation, as described in Section 5.

4. RepuNet: An adaptive reputation system to mitigate attacks in DFL

This section presents RepuNet, a decentralized, adaptive reputation system designed to detect and mitigate attacks in DFL. RepuNet continuously evaluates the behavior of participating nodes using a set of observable metrics and adjusts their influence on model aggregation based on their reputation. This section details the reputation scoring mechanism, the metrics used, the step-by-step operational flow, the reputation update process, the exclusion and recovery criteria, and the practical integration of the mechanism into the Nebula experimental platform.

4.1. Reputation system foundations

RepuNet is designed for fully DFL environments with the goal of evaluating node behavior to detect and mitigate malicious activity while preserving contribution quality and minimizing overhead. Each node autonomously decides to accept, weight, or exclude incoming models. Its design emphasizes training efficiency and system stability, avoiding bottlenecks or mechanisms that could hinder global convergence. This enables a robust and resilient collaborative environment, even in the presence of anomalous behavior.

4.2. Decentralized architecture and operation

RepuNet is modular and decentralized. Each node in the federated network includes an autonomous reputation component that evaluates the behavior of its neighbors during each training round. This evaluation determines whether received models should be filtered or adjusted before aggregation, depending on the sender's reputation. Fig. 1 shows two main components. On the left is the standard local training flow: local model training, model broadcasting, receiving external models, reputation-based filtering, and aggregation. On the right, the internal RepuNet module computes a reputation score for each neighbor based on observable behavioral metrics. The outcome of this evaluation guides the filtering and adjustment decisions. Upon receiving external models, the node calls the RepuNet module. This module evaluates each neighbor based on metrics such as model similarity, the fraction of parameters changed, the delay in model arrival, and the incoming message flow. These metrics are processed and weighted, either statically or adaptively, to produce a reputation score. Feedback from other nodes is also aggregated to enhance the evaluation. Reputations are stored locally and then disseminated to neighbors in each protocol round. The updated reputation influences the filtering decision: if the reputation is

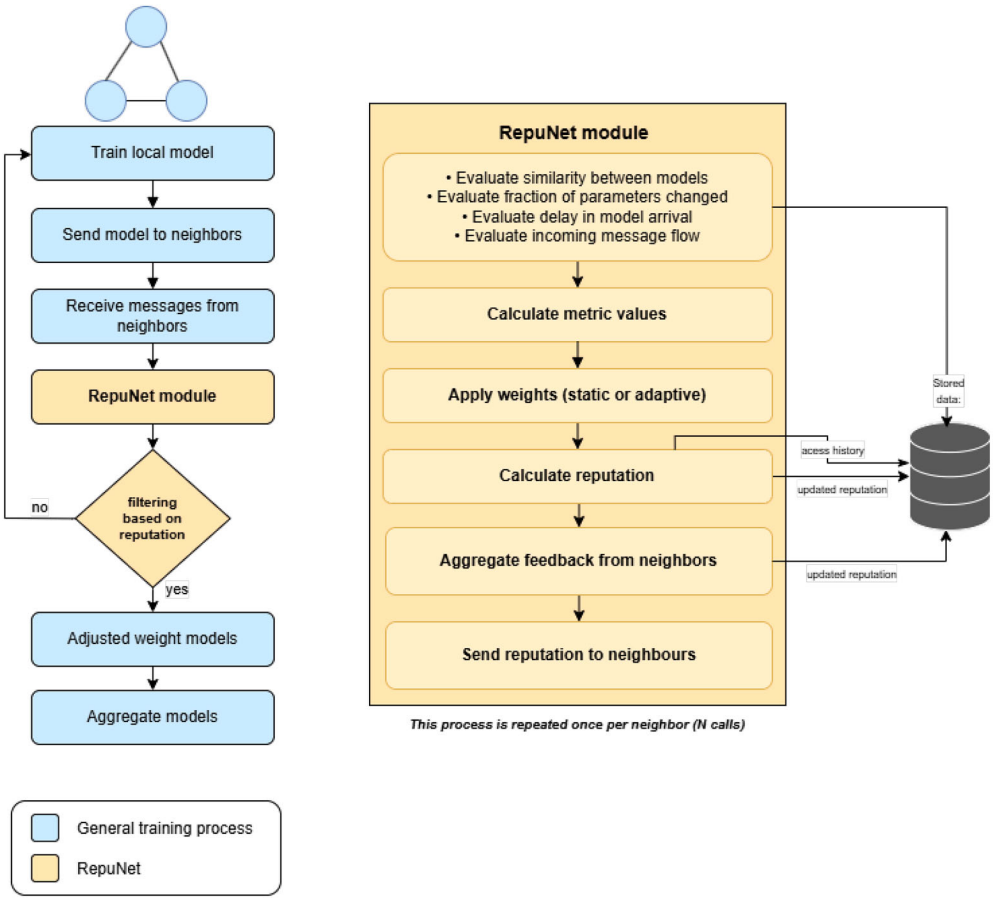


Fig. 1. Interaction between a federated node and the RepuNet module, showing the flow of training, reputation evaluation, and aggregation.

below a given threshold, the model is discarded; otherwise, it is incorporated with a weight proportional to its reputation. The evaluation is performed once per neighbor per round.

The complete operational flow is summarized below:

1. **Local training:** The node updates its model using local private data.
2. **Model broadcast:** The model is shared with neighboring nodes.
3. **Message reception:** Models are received from neighbors.
4. **Reputation evaluation:** RepuNet evaluates each neighbor using local metrics and aggregated feedback.
5. **Filtering decision:** If a neighbor's reputation is too low, the model is discarded.
6. **Adjusted weighting:** Accepted models are weighted based on the sender's reputation.
7. **Aggregation:** Reputation-weighted models are aggregated with the local model.
8. **Reputation dissemination:** Updated reputation scores are stored and sent to all neighbors.

Unlike binary filters, RepuNet adjusts each model's aggregation weight. This adaptive integration allows the contribution of each node to evolve dynamically based on behavioral history. The next sections detail the computation of reputation and the metrics involved. The operational logic can be formalized through Algorithm 1, which summarizes the reputation-guided federated training cycle.

4.3. Metrics used in RepuNet

RepuNet relies on four primary metrics to evaluate the behavior of neighbors in a DFL network. These metrics are divided into two categories: *model quality* and *communication behavior*. Each metric returns a

Algorithm 1 Federated round with reputation-guided adaptive aggregation.

COMMUNICATION_ROUND()
  TrainLocalModel()
  SendModelToNeighbors()
  $messages \leftarrow$ **ReceiveMessagesFromNeighbors()**
for each $message$ in $messages$ **do**
  $rep \leftarrow$ **GetReputation**($message.sender$)
  if IsReputationSufficient(rep) **then**
    $weight \leftarrow$ **ComputeWeightFromReputation**(rep)
    StoreAcceptedModel($message.model$, $weight$)
  end if
end for
AggregateAcceptedModels()

UPDATE_REPUTATION()
for each $neighbor$ **do**
  $metrics \leftarrow$ **CalculateMetrics**($neighbor$)
  $score \leftarrow \sum_j weights[j] \cdot NormalizeMetric(metrics[j])$
  $history \leftarrow$ **GetReputationHistory**($neighbor$)
  $rep \leftarrow \sum_{r \in history} \omega^{(r)} \cdot rep[r] + \omega^{(t)} \cdot score$
  if feedback available **then**
    $avg\_fb \leftarrow$ average reputation received from neighbors
    $rep \leftarrow \eta \cdot rep + (1 - \eta) \cdot avg\_fb$
  end if
  StoreReputation($neighbor$, rep)
  SendReputationToNeighbor($neighbor$, rep)
end for

normalized value between 0 and 1, where 1 indicates optimal behavior. The system applies soft penalties to deviations, and the metrics are combined with dynamic weights to generate the reputation score.

4.3.1. Model similarity

This metric assesses the alignment between the local model and the one received. Four classic distance measures are used: cosine, Euclidean, Manhattan, and Pearson correlation. The final similarity is a weighted average:

$$\text{similarity} = \sum_k \gamma_k \cdot S_k \tag{1}$$

where S_k represents each individual measure and γ_k its weight. Low values indicate anomalous deviations in model parameters. where S_k represents the weight assigned to each metric. In this work, we employ uniform weights ($S_k = 0.25$ for all k), justified by the following rationale: First, the four distance measures are designed to capture orthogonal aspects of model similarity-cosine similarity measures directional alignment, Euclidean and Manhattan distances capture absolute parameter magnitudes, and Pearson correlation reflects linear statistical relationships. Since these metrics are complementary and non-redundant, assigning equal weights avoids introducing unjustified bias toward any particular dimension. Second, uniform weighting follows the principle of maximum entropy in the absence of prior knowledge about the relative importance of each similarity aspect, thereby maximizing robustness against diverse attack strategies. This symmetric approach ensures reproducibility and prevents adversarial exploitation of weighted schemes. By treating all similarity measures equally, the composite metric achieves balanced anomaly detection across heterogeneous parameter deviations, whether they manifest in orientation, magnitude, or statistical correlation.

4.3.2. Fraction of parameters changed

RepuNet computes this metric by maintaining a history of change fraction values (f) and their associated thresholds (t), from which the mean and standard deviation are derived:

$$\mu_f = \frac{1}{N}\sum_{i=0}^{N} f_i, \qquad \sigma_f = \sqrt{\frac{1}{N}\sum_{i=0}^{N}(f_i - \mu_f)^2} \tag{2}$$

$$\mu_t = \frac{1}{N}\sum_{i=0}^{N} t_i, \qquad \sigma_t = \sqrt{\frac{1}{N}\sum_{i=0}^{N}(t_i - \mu_t)^2} \tag{3}$$

Based on these statistics, RepuNet defines acceptable upper bounds for both metrics using a tolerance margin slightly above the mean plus one standard deviation (typically 5%-10%). This margin captures the natural variability of benign updates while allowing the detection of significant deviations as anomalies. Practically, suppose the current value of a metric exceeds its corresponding upper bound, which is computed as the mean plus the product of the standard deviation and the tolerance factor. In that case, it is marked as anomalous, and a proportional penalty is then calculated:

$$P = \frac{|f_{\text{current}} - \mu_f|}{\mu_f} \tag{4}$$

This penalty is transformed into a smoothed score using a sigmoid function:

$$S = 1 - \left(\frac{1}{1 + e^{-P}}\right) \tag{5}$$

If no anomaly is detected, the score is perfect: $S = 1.0$. The scores for the change fraction (S_f) and threshold (S_t) are combined with equal weights:

$$f_{\text{final}} = 0.5 \cdot S_f + 0.5 \cdot S_t \tag{6}$$

To avoid overpenalizing isolated variations, a temporal smoothing mechanism is applied:

$$f^{(t)}_{\text{final}} = \lambda^{(t)} \cdot f^{(t)}_{\text{actual}} + (1 - \lambda^{(t)}) \cdot f^{(t-1)}_{\text{final}} \tag{7}$$

where $\lambda^{(t)} = 0.2$ is a sensitivity coefficient that allocates 20% weight to the current anomaly score and 80% to the historical average. This conservative weighting prevents overreaction to isolated parameter variations while remaining sensitive to sustained deviations, which is particularly important in federated learning, where stochastic local training naturally introduces fluctuations.

Finally, if a node does not receive a model from a neighbor during an iteration, this metric is automatically penalized by reducing the score by 50% from its previous value.

4.3.3. Model arrival latency

This metric evaluates how promptly a neighboring node delivers its model in each communication round. Its goal is to detect **anomalous delays** that may affect node synchronization and aggregation stability.

The computation begins by logging the model arrival time t_{arrival}, compared to the start time of the round. Depending on the temporal origin of the model, latency is defined as:

$$\text{latency} = \begin{cases} t_{\text{arrival}} - t_{\text{start\_current\_round}} & \text{current round model} \\ t_{\text{arrival}} - t_{\text{start\_model\_round}} & \text{previous round model} \end{cases} \tag{8}$$

Based on past observations, the historical latency average is computed as:

$$\overline{\text{latency}} = \frac{1}{N}\sum_{i=1}^{N} \text{latency}_i \tag{9}$$

When the attack begins from the first iteration, round 0 is used as the baseline, and the tolerance threshold is set at 150% of the average latency. The deviation is calculated as the difference between the current latency and the historical average latency. The final score is assigned as follows:

$$\text{score} = \begin{cases} 1.0 & \text{if within threshold (150\%)} \\ \dfrac{1}{1 + \exp\left(\frac{|\Delta t|}{\tau^{(t)}}\right)} & \text{if above} \end{cases} \tag{10}$$

where Δt is the observed deviation from the historical mean and $\tau^{(t)}$ is a tunable temporal tolerance parameter; the acceptance threshold is defined as 150% of the historical average latency, above which the score is progressively penalized. To smooth inter-round variation, an exponential update is applied:

$$\bar{s}^{(t)}_{\text{lat}} = \mu^{(t)} \cdot s^{(t)}_{\text{lat}} + (1 - \mu^{(t)}) \cdot \bar{s}^{(t-1)}_{\text{lat}} \tag{11}$$

In early rounds, where history is limited, a bootstrapping penalty is introduced:

$$\bar{s}^{(t)}_{\text{lat}} = s^{(t)}_{\text{lat}} \cdot (1 - \delta^{(t)}), \quad \delta^{(t)} \in [0, 1], \text{ typically } \delta^{(t)} = 0.05 \tag{12}$$

If no model is received from a node during a round, its score is reduced by 50% from the previous round.

This metric helps detect both intentional delays and *lack of participation*, thereby preserving temporal coherence within the federation.

4.3.4. Incoming message flow

This metric evaluates a node's communicative activity by comparing the number of messages sent to a specific neighbor in the current round with the collective behavior in the previous round. Its goal is to detect anomalies, such as excessive or persistent overcommunication, typical of flooding attacks.

For each node pair (a, b), the number of messages sent in round r is denoted $m_r^{(a,b)}$, and the previous round $(r-1)$ message counts across all pairs (i, j) are collected. From this, the 25th percentile P_{25} is computed as a reference, along with standard deviation σ_{r-1} and mean μ_{r-1}.

The relative increase is defined as:

$$\text{rel\_incr} = \max\left(\frac{m_r^{(a,b)} - P_{25}}{P_{25}},\ 0\right) \tag{13}$$

Table 2
Summary of metrics used in reputation computation.

Metric	Type	Purpose	Symbol	Range	Penalty trigger	Normalization
Model similarity	Model quality	Align local and received models	S	[0, 1]	Low similarity	Weighted average (cosine, Euclidean, Manhattan, Pearson)
Fraction of parameters changed	Model quality	Detect abrupt model changes	F	[0, 1]	High deviation from past	Sigmoid penalty on deviation
Model arrival latency	Communication	Penalize delayed models	L	[0, 1]	Above historical latency	Sigmoid based on delay deviation
Incoming message flow	Communication	Detect flooding behavior	M	[0, 1]	High message volume	Exponential penalty vs. dynamic margin

A dynamic tolerance margin is calculated based on variability:

$$\text{dynamic\_margin} = \frac{\sigma_{r-1} + 1}{\log(1 + P_{25}) + 1} \tag{14}$$

If the relative increase exceeds the margin, a smoothed logarithmic exponential penalty is applied:

$$s^{(r)}_{\text{msg}} \leftarrow s^{(r)}_{\text{msg}} \cdot \exp\left(-\left(\frac{\log\left(1 + \text{rel\_incr} - \text{dynamic\_margin}\right)}{\log\left(1 + \text{dynamic\_margin}\right) + \varepsilon}\right)^2\right) \tag{15}$$

If the node also exceeds the global mean:

$$\text{amplification} = 1 + \frac{increase}{\text{-}}\text{mean}\mu_{r-1} + \varepsilon \tag{16}$$

$$s^{(r)}_{\text{msg}} \leftarrow s^{(r)}_{\text{msg}} \cdot \exp\left(-(\text{extra\_penalty} \cdot \text{amplification})^2\right) \tag{17}$$

where increase_mean adjusts automatically based on the system phase.

A per-node and per-neighbor temporal history is maintained. Repeated low scores across rounds trigger additional multiplicative penalties. To avoid permanent exclusion, a minimum adaptive bound is enforced:

$$s^{(r)}_{\text{msg}} \leftarrow \max\left(s^{(r)}_{\text{msg}}, f_{\min}(r, a, b)\right) \tag{18}$$

Values are smoothed with a weighted average of up to three prior rounds. This formulation detects and penalizes meaningful communication deviations in a proportional, dynamic manner, without requiring fixed thresholds.

These four metrics form the foundation of the RepuNet reputation system and are summarized in Table 2.

4.4. Weight assignment and reputation computation

RepuNet assigns dynamic weights and computes the reputation of each neighbor in two main steps: (i) evaluation of the current round, and (ii) combination with historical scores.

Definitions of symbols. For each node i evaluating a neighbor B in round t, we define:

- $m_j^{(t)} \in [0, 1]$: normalized value of metric j for neighbor B in the current round.
- μ_j: historical mean of metric j.
- $d_j^{(t)} = |m_j^{(t)} - \mu_j|$: absolute deviation of metric j from its historical mean.
- $D^{(t)} = \sum_k d_k^{(t)}$: sum of deviations across all metrics for this round.
- $w_j^{(t)}$: normalized weight assigned to metric j for this round.
- $S_B^{(t)}$: intermediate reputation score of neighbor B computed by node i based on the current round.
- $\mathcal{H}_B^{(t)}$: history of past reputation scores for neighbor B.
- $\omega^{(r)}$ and $\omega^{(t)}$: weights for historical and current scores ($\sum \omega = 1$).
- $R_B^{(t)}$: reputation of neighbor B according to node i after incorporating history.

Step 1: Score computation for the current round. Node i evaluates neighbor B using the metrics for the current round. Deviations are computed as:

$$d_j^{(t)} = \left|m_j^{(t)} - \mu_j\right| \tag{19}$$

The sum of deviations is:

$$D^{(t)} = \sum_k d_k^{(t)} \tag{20}$$

Normalized weights for each metric are then calculated as:

$$w_j^{(t)} = \begin{cases} \frac{d_j^{(t)}}{D^{(t)}}, & \text{if } D^{(t)} \neq 0 \\ \text{random normalized weights}, & \text{otherwise.} \end{cases} \tag{21}$$

The intermediate reputation score for this round is:

$$S_B^{(t)} = \sum_j w_j^{(t)} \cdot m_j^{(t)} \tag{22}$$

This score reflects the reputation of neighbor B as seen by node i, based solely on the metrics and weights assigned by i.

Step 2: Incorporation of historical reputation. Node i then combines the intermediate score with the historical reputation of neighbor B:

$$R_B^{(t)} = \sum_{r \in \mathcal{H}_B^{(t)}} \omega^{(r)} \cdot R_B^{(r)} + \omega^{(t)} \cdot S_B^{(t)} \tag{23}$$

This ensures that the reputation reflects both past behavior and current evaluations, allowing nodes to gradually recover reputation as they improve their behavior.

Note on neighbor feedback. If reputation feedback from neighbors is enabled, the final reputation $\tilde{R}_B^{(t)}$ will be computed as described in Section 4.6, avoiding repetition of the formula in this section.

4.5. Aggregation, exclusion, and recovery control

Once the final reputation of each node has been computed, RepuNet uses this value to decide whether to include or exclude each node's model in the aggregation process. If a neighbor's reputation falls below a predefined trust threshold, their model is automatically discarded for that round. This exclusion prevents anomalous, malicious, or inconsistent behaviors from influencing the global model. It is important to note that exclusion does not imply disconnection or blocking of the node in the system: the penalty only applies to the use of its model, allowing continued participation in subsequent rounds and re-evaluation. This ensures the architecture remains open and tolerant toward nodes that may recover their behavior after temporary penalties. The exclusion threshold is configurable and can be adjusted to the environment or attack type. In typical configurations, reputation values below 0.6 result in model rejection, whereas higher reputations allow the node's contribution to be weighted accordingly. Furthermore, reputation is evaluated

and updated in every round. If a node improves its behavior and its metrics align with the expected pattern, its reputation gradually increases, enabling natural, progressive reintegration into aggregation. This recovery property avoids permanent penalties for transient issues or isolated failures. Overall, this control mechanism provides a flexible, adaptive, and resilient system that acts as a dynamic filter against untrustworthy or malicious contributions while encouraging the reintegration of rehabilitated nodes.

4.6. Distributed feedback and robustness

In addition to local evaluation, RepuNet allows each node to incorporate reputation feedback from trusted neighbors. This mechanism reinforces individual assessments, particularly when a node has limited observations or its perception is distorted by network conditions. Each node may receive reputation scores issued by its neighbors in the current round. The feedback is incorporated through a weighted combination of the local evaluation $R_i^{(t)}$ and the average of the received evaluations:

$$\tilde{R}_i^{(t)} = \eta \cdot R_i^{(t)} + (1-\eta) \cdot \text{mean}\Big(\{ R_{j \to i}^{(t)} \} \Big) \tag{24}$$

where $R_{j \to i}^{(t)}$ are the reputations reported by neighbors, and $\eta \in [0, 1]$ controls the trust in the internal assessment versus neighbor feedback. Since only nodes with an active reputation module can issue evaluations, all received scores originate from valid participants, preventing manipulation. This decentralized consensus approach smooths isolated evaluation errors, mitigates local fluctuations, and reinforces robustness against attacks or biased perceptions. In subsequent rounds, the system repeats the same evaluation cycle: metric computation, dynamic weight assignment, historical reputation update, incorporation of neighbor feedback, and dissemination of the final reputation. This continuous mechanism ensures the federated system remains resilient while supporting the reintegration of nodes that recover reliable behavior.

4.7. Integration of RepuNet into the nebula platform

RepuNet has been implemented as a modular, autonomous, and platform-independent component. For experimental evaluation, it was integrated into the *Nebula* platform, which supports decentralized FL scenarios and includes real-time visualization via TensorBoard [16]. The integration enables testing across topologies and attack types.

Nebula's architecture consists of three main layers:

- **Frontend:** a graphical interface where users can:
  - Activate or deactivate RepuNet.
  - Select evaluation metrics and their weighting scheme.
  - Configure parameters such as the exclusion threshold, feedback weight η, and temporal smoothing.
- **Controller:** an intermediate layer that propagates configurations to all nodes, ensuring consistency across rounds.
- **Core:** the execution layer for each node, which handles:
  - Local training on private data.
  - Model exchange with neighbors.
  - Metric evaluation and reputation updates.
  - Weighted aggregation based on reputation.

Each node maintains its own history of metrics and reputations, enabling smoothing, recurrence detection, and adaptive behavior. The integrated visualization tools help track the evolution of each metric, monitor node reputation across rounds, and observe the influence of distributed feedback (Fig. 2).

5. RepuNet evaluation

This section evaluates RepuNet in adversarial DFL environments. The objective is to assess its effectiveness in detecting and penalizing malicious nodes, preserving model performance and training stability. The experiments vary the proportion of attackers and the timing of activation, analyzing the evolution of reputation scores and their effect on aggregation, with and without defense.

5.1. Experimental environment

Experiments were run on Nebula, which emulates DFL via virtual scenarios. Nodes train, evaluate, and aggregate locally without a central server. The entire federation runs as independent processes on a single machine, enabling full control and real-time monitoring. Table 3 summarizes the general setup parameters, including datasets, topologies, hardware configuration, and training schedule used across all experiments.

We emulated model poisoning, delay, and flooding attacks to test robustness. For each, the experimental task and measurement objective are clearly defined in Table 4.

5.2. Model poisoning attack

This attack degrades the global model by injecting malicious updates that differ significantly from honest contributions and may hinder convergence if undetected. RepuNet evaluates each neighbor before aggregation based on behavioral metrics. Experiments were conducted on the MNIST and CIFAR-10 datasets across different topologies, attacker proportions, and data heterogeneity levels. Table 5 summarizes the configuration of all poisoning scenarios, including early and intermittent variants. Each scenario is labeled with a short identifier (e.g., *Base*, *2.1*, *6.1*) that is used consistently across the results and discussion.

Table 6 reports the F1-score for each scenario with and without RepuNet. Scores are measured at rounds 8 and 11, respectively. The difference ΔF_1 quantifies the improvement, and the last column categorizes the observed impact.

Fig. 3 provides a visual summary of the impact levels across all scenario groups. Subfigures organize experiments by activation type or configuration and highlight thresholds for low, medium, and high impact levels. Three representative scenarios illustrate RepuNet's response to poisoning attacks. These experiments use the MNIST dataset. Each case includes two plots: one showing the average reputation of all nodes, and another showing only malicious nodes. The results are shown in Fig. 4, which illustrates how reputation evolves in response to adversarial behavior under different topologies and attacker proportions.

Scenario **Base** represents a fully connected topology with 30% malicious nodes. RepuNet responds quickly after the attack is triggered in round 7, penalizing adversarial behavior. The reputation of honest nodes remains high, while that of malicious nodes drops noticeably. Nodes 192.168.51.10:45009 and 192.168.51.11:45010 were selected for illustration. Scenario **3.2** uses a random topology and maintains the same proportion of attackers. Greater variability in reputation is observed due to the network's sparse connectivity, which reduces the influence of malicious nodes. The drop in reputation begins in round 7, confirming RepuNet's ability to adapt in less structured environments. Nodes 192.168.51.2:45001 and 192.168.51.9:45008 are shown. Scenario **2.3** corresponds to a more adversarial environment, with a fully connected topology and 60% malicious nodes. Despite the increased pressure, RepuNet continues to penalize malicious behavior consistently. The Dirichlet parameter $\alpha = 0.5$ used for data partitioning has a limited effect in this setting. Nodes 192.168.51.4:45003 and 192.168.51.9:45008 were selected for observation. To illustrate the internal adjustment process, Table 7 shows how node 10 tracks a malicious neighbor across rounds. In rounds 7 and 8, after the attack starts, the model similarity and fraction of parameters changed drop significantly. This leads to an automatic reweighting, increasing the importance of these more discriminative metrics. The system adapts to contextual anomalies, penalizing malicious behavior without manual intervention.

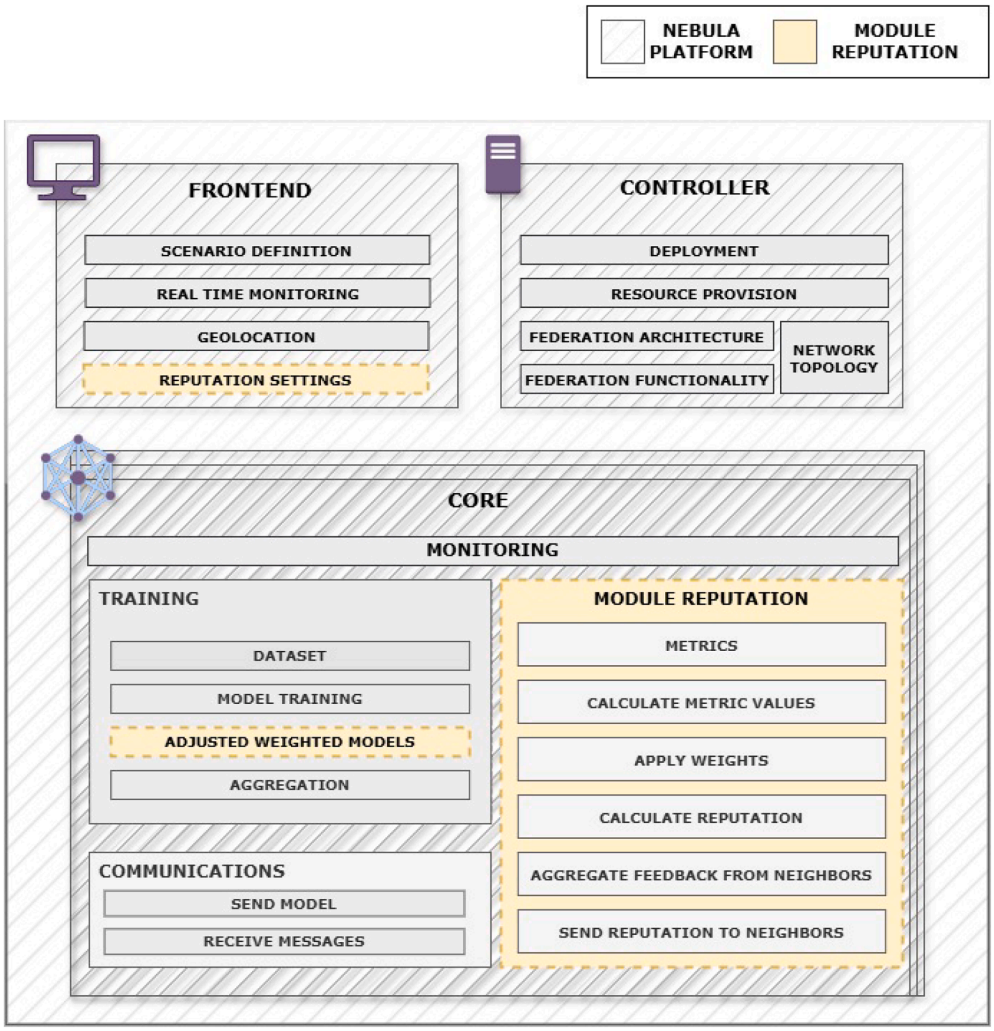


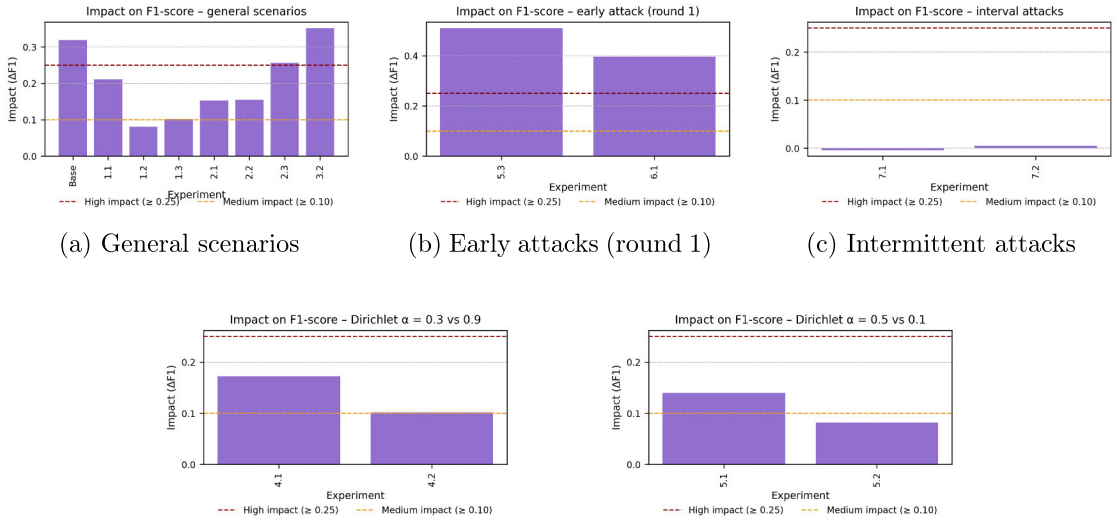
Fig. 2. Integration of the RepuNet reputation system into the Nebula platform architecture.

Table 3
Summary of experimental setup parameters.

Parameter	Description
Platform	Nebula (Decentralized FL emulation platform)
Datasets	MNIST, CIFAR-10
Nodes per federation	10–25 nodes
Topology types	Fully connected, Random
Data partition	Dirichlet (Non-IID), $\alpha \in \{0.1, 0.3, 0.5, 0.9\}$
Local epochs	MNIST: 1 epoch/round; CIFAR-10: 10 epochs/round
Aggregation algorithm	FedAvg, Krum, TrimmedMean
Hardware specs	Intel Xeon CPU E5-2697, 62 GB RAM, Ubuntu 22.04
Number of rounds	20 communication rounds
Timeout for aggregation	30–60 seconds
Metrics used	Model Similarity, Fraction of Parameters Changed, Arrival Latency, Incoming Message Flow

Table 4
Attack types, tasks, and measurement objectives.

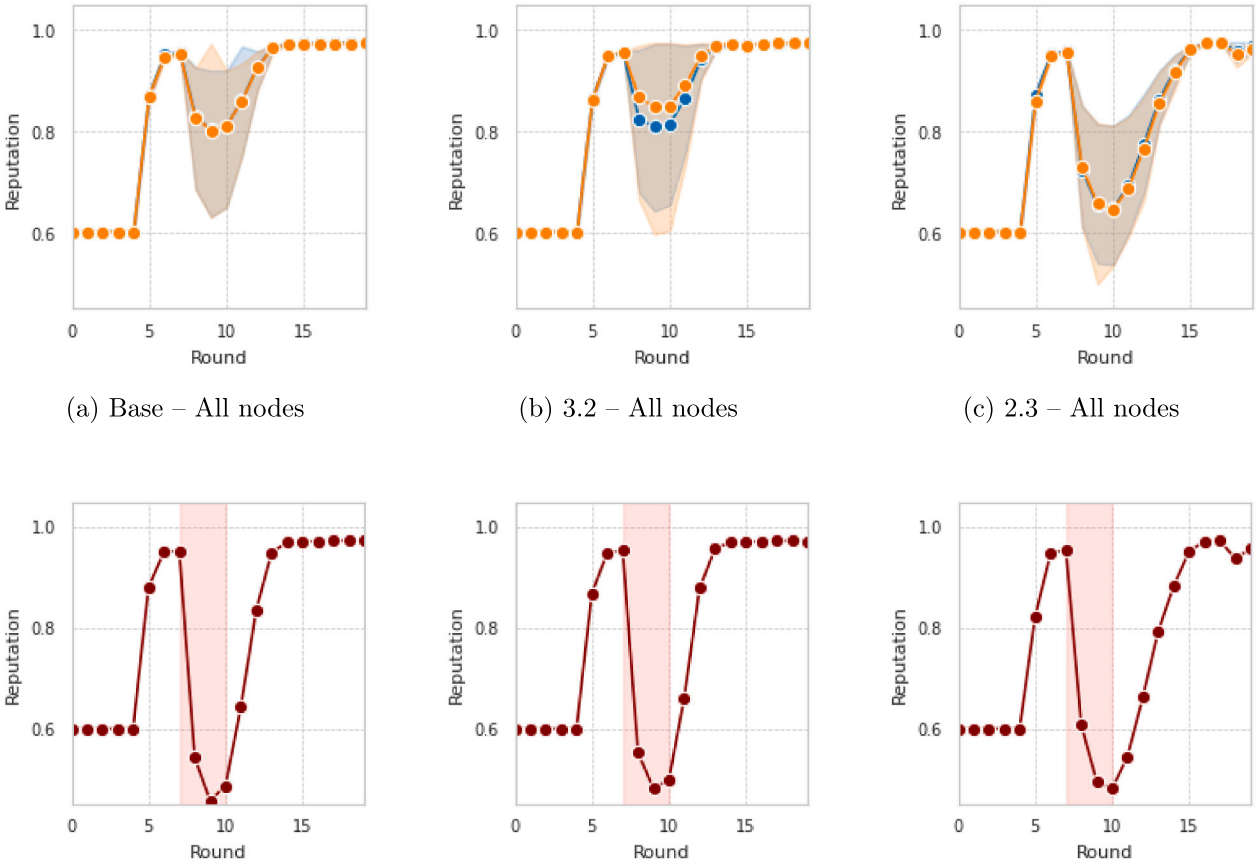
Attack Type	Task	Aggregation Algorithm	Measurement Objective
Model Poisoning	Image classification (MNIST, CIFAR-10)	FedAvg, Krum, TrimmedMean	Evaluate degradation of global model accuracy (F1-score) due to manipulated updates.
Delay attack	Image classification (MNIST)	FedAvg	Assess synchronization issues and increased aggregation latency caused by delayed submissions.
Flooding Attack	Image classification (MNIST)	FedAvg	Measure communication efficiency degradation and CPU overhead caused by excessive messaging.



(a) General scenarios (b) Early attacks (round 1) (c) Intermittent attacks

(d) Dirichlet α variation (4.1 vs 4.2) (e) Dirichlet α variation (5.1 vs 5.2)

Fig. 3. Impact of the RepuNet reputation system in various *model poisoning* attack scenarios. (a) General scenarios, (b) attacks from round 1, (c) intermittent attacks, (d–e) sensitivity to the heterogeneity parameter α of the Dirichlet distribution. Horizontal lines mark the thresholds for high impact (≥ 0.25) and medium impact (≥ 0.10).



(a) Base – All nodes (b) 3.2 – All nodes (c) 2.3 – All nodes

(d) Base – Malicious nodes (e) 3.2 – Malicious nodes (f) 2.3 – Malicious nodes

Fig. 4. Reputation evolution in three scenarios with *model poisoning* attacks. Top row: average reputation of all federation nodes. Bottom row: average reputation of malicious nodes.

Table 5
Model poisoning attack scenarios: configuration summary.

Scenario Group	Topology	Nº nodes	Malicious Nodes	Dirichlet α
Base / 1.x / 2.x	Fully	10	30–60%	0.5
3.1	Ring	10	30%	0.5
3.2	Random	10	30%	0.5
4.1 / 4.2	Fully	10	30%	0.3 / 0.9
5.1 / 5.2 / 5.3	Fully	10	30%	CIFAR10: 0.5 / 0.1
6.1	Fully	10	30%	0.5 (early)
7.1 / 7.2	Fully	10	30%	0.5 (interval 2/3)

Table 6
F1-score comparison across model poisoning scenarios with and without RepuNet.

Scenario	F1 w/rep (r8)	F1 w/o rep (r11)	ΔF_1	Impact
Base	0.6800	0.3610	0.3190	High
1.1	0.6340	0.4226	0.2114	Medium
1.2	0.6670	0.5868	0.0802	Low
1.3	0.6447	0.5430	0.1017	Medium
2.1	0.5972	0.4444	0.1528	Medium
2.2	0.5051	0.3506	0.1545	Medium
2.3	0.4143	0.1575	0.2568	High
3.1	0.6821	0.4017	0.2804	High
3.2	0.6744	0.3231	0.3513	High
4.1	0.6736	0.5012	0.1724	Medium
4.2	0.6787	0.5763	0.1024	Medium
5.1	0.5360	0.3965	0.1395	Medium
5.2	0.5122	0.4301	0.0821	Low
5.3	0.5166	0.0720	0.4446	High
6.1	0.6191	0.2225	0.3966	High
7.1	0.6796	0.6833	-0.0037	Low
7.2	0.6879	0.6825	0.0054	Low

Table 7
Metric and weight evolution for a malicious neighbor (node 10).

Metric	Round 5	Round 6	Round 7	Round 8
Weight: similarity	0.25	0.24	0.41	0.43
Weight: fraction	0.22	0.22	0.38	0.39
Weight: latency	0.28	0.28	0.11	0.10
Weight: messages	0.25	0.26	0.10	0.08

Baseline Robust Aggregation Methods under Model Poisoning. In addition to the default FedAvg configuration, we extend the poisoning evaluation by incorporating representative Byzantine-resilient decentralized aggregation mechanisms. Specifically, we evaluate Krum and Trimmed Mean, two widely adopted robust aggregation strategies designed to mitigate the influence of anomalous or malicious model updates.

These mechanisms approximate the type of statistical filtering defenses proposed in Byzantine-resilient decentralized learning approaches [17,18], where robustness is achieved through round-based gradient or parameter selection rules.

To ensure comparability, we maintain the similar experimental setting described in Section 5.2: CIFAR-10 dataset, 10 nodes in a fully connected topology, 20 communication rounds, 2 local epochs per round, Dirichlet non-IID partition with $\alpha = 0.5$, and 30% malicious nodes under model poisoning. The attack is activated at round 5 and remains active until round 10.

For these experiments, the reputation mechanism is restricted to model-based metrics only (model similarity and fraction of parameters changed), while communication-based metrics (arrival latency and message flow) are intentionally disabled. Since the evaluated attack does not introduce communication anomalies, including these metrics could artificially increase the reputation of malicious nodes. Therefore, we isolate the analysis to model-behavior indicators. This configuration highlights the modular and configurable nature of the reputation component, allowing targeted evaluation depending on the attack type.

For each aggregation strategy (Krum and Trimmed Mean), we evaluate three scenarios: (i) no attack without reputation, (ii) poisoning without reputation, and (iii) poisoning with RepuNet enabled. To provide a detailed temporal evaluation, we analyze the evolution of F1-score and average reputation across three phases: before the attack (rounds 1-4), during the attack (rounds 5-10), and after the attack (rounds 11-20), with poisoning activated between rounds 5 and 10. The same temporal windows are applied to the no-attack scenario to ensure comparability. This segmentation enables us to assess the degradation introduced by poisoning, the recovery capacity of the system, and whether reputation accumulated prior to the attack leads to false positives or incorrect penalization of benign nodes. When reputation is disabled, the corresponding reputation metrics are left empty in Table 8.

In addition to the temporal F1 analysis, we further quantify the structural behavioral separation induced by the reputation mechanism under active attack conditions. Specifically, we compute the performance gap between benign and malicious nodes, defined as the difference between their mean F1-scores, as well as the global variance of F1 across all participants at round 8 (mid-attack). Since behavioral differentiation is only meaningful when adversarial nodes are present, structural metrics are interpreted exclusively during the attack phase. This analysis allows us to evaluate whether reputation enhances behavioral differentiation beyond global predictive performance. The results are summarized in Table 9.

Stronger and Structured Poisoning Attacks. As defined in Section 3, we evaluate two structured model poisoning strategies: Signed Neuron Remapping (SNR) and GLL Neuro Inversion. These attacks follow the class of coordinated and stealthy poisoning strategies described in prior literature on Byzantine-resilient federated learning [19,20], where adversarial updates preserve global magnitude properties while introducing structured directional bias. Such manipulations challenge magnitude-based and distance-based aggregation defenses by avoiding extreme parameter outliers.

Signed Neuron Remapping (SNR). In this strategy, malicious nodes apply a neuron-level permutation toward cosine-dissimilar representations followed by sign inversion of the associated weights. This operation preserves norm consistency while disrupting internal representation alignment across layers. Unlike simple random perturbations, SNR maintains numerical coherence while introducing structural inconsistencies that degrade aggregation stability.

GLL Neuro Inversion. In this configuration, malicious nodes perform structured inversion over selected neuron groups, introducing coordinated directional bias in the update while maintaining global magnitude stability. This manipulation degrades gradient alignment during aggregation without producing extreme parameter outliers.

The experimental configuration remains consistent with the setup described in Table 3, including 10 nodes, 30% malicious participants, MNIST dataset, Dirichlet $\alpha = 0.5$, 20 communication rounds, and attack activation between rounds 5 and 10.

Table 10 summarizes the temporal performance of the structured poisoning strategies under three configurations: clean baseline (no attack, no reputation), attack without reputation, and attack with reputation enabled. Results are reported in terms of F1-score before (B), during (D), and after (A) the adversarial interval, together with the mean reputation values across the federation. Under GLL Neuro Inversion, standard FedAvg exhibits severe degradation during the adversarial phase and fails to recover after the attack, whereas enabling RepuNet mitigates the degradation and restores stable post-attack convergence, with decreasing mean reputation values reflecting progressive penalization of malicious participants. In the case of Signed Neuron Remapping (SNR), standard aggregation shows moderate degradation and limited post-attack improvement, while reputation-based filtering improves performance both during and after the attack, indicating effective attenuation of structurally inconsistent updates and improved federation stabilization.

Table 8
Baseline robust aggregation under model poisoning (B = before, D = during, A = after).

Aggregation	Attack	Reputation	F1 (B)	F1 (D)	F1 (A)	Mean Rep. (B)	Mean Rep. (D)	Mean Rep. (A)
Krum	No	No	0.1764	0.2607	0.3064	–	–	–
Krum	Yes	No	0.1819	0.2152	0.3089	–	–	–
Krum	Yes	Yes	0.1729	0.2162	0.2707	0.8279	0.7565	0.8844
Trimmed Mean	No	No	0.3729	0.6014	0.6793	–	–	–
Trimmed Mean	Yes	No	0.3703	0.1493	0.3277	–	–	–
Trimmed Mean	Yes	Yes	0.2480	0.2623	0.4046	0.8692	0.7402	0.8670

Table 9
Behavioral separation at round 8 (During Attack Phase).

Aggregation	Attack	Reputation	F1 GAP	F1 Variance
Krum	Yes	No	0.2700	0.0124
Krum	Yes	Yes	0.2845	0.0196
Trimmed Mean	Yes	No	0.0557	0.00067
Trimmed Mean	Yes	Yes	0.3106	0.0250

Table 11 further clarifies the structural impact of reputation during the adversarial interval. For GLL Neuro Inversion, standard aggregation leads to global performance degradation with limited structural separation at round 8. When reputation is enabled, benign nodes maintain high performance while malicious nodes collapse to near-zero F1, resulting in a substantially larger GAP and increased variance. This behavior reflects effective structural isolation rather than instability, as the dispersion is driven by clear differentiation between trusted and penalized participants.

Under Signed Neuron Remapping (SNR), the structural pattern differs. While variance increases when reputation is enabled, the GAP decreases compared to the configuration without reputation. This indicates that, rather than generating abrupt polarization, reputation progressively attenuates malicious influence while preserving overall stability. The resulting dispersion reflects controlled differentiation dynamics instead of extreme behavioral collapse.

The pronounced structural separation observed in Table 11 motivates a more detailed analysis of how similarity-based differentiation operates under structured manipulation. Since both GLL and SNR preserve global magnitude while altering internal representation alignment, detecting anomalies requires distribution-aware clustering rather than fixed-threshold filtering.

Let $\{S_1, S_2, \dots, S_n\}$ denote the similarity scores computed for all neighbors at a given round. These values are sorted in ascending order,

$$S_{(1)} \le S_{(2)} \le \cdots \le S_{(n)} \tag{25}$$

and the gap between consecutive ordered values is defined as

$$g_k = S_{(k+1)} - S_{(k)}, \quad k = 1, \dots, n-1. \tag{26}$$

The maximum gap

$$g^* = \max_k g_k \tag{27}$$

identifies the most significant structural separation within the similarity distribution. If this gap satisfies predefined validation conditions (minimum magnitude, dominance over remaining gaps, and bounded cluster size), nodes in the lower cluster

$$\{S_{(1)}, \dots, S_{(k^*)}\} \tag{28}$$

are marked as structurally inconsistent in the current round.

Persistent assignment to this cluster increments a violation counter v_i, progressively reducing similarity contribution and eventually triggering exclusion once $v_i \ge V_{\max}$. This adaptive mechanism explains the amplified structural differentiation observed under structured poisoning: benign nodes remain tightly clustered, while recurrently inconsistent participants are sharply separated.

5.3. Delay attack

The *delay* attack disrupts the federated process by intentionally delaying the transmission of model updates. While it does not alter model content, it exploits the temporal dimension to desynchronize nodes, degrade convergence, and stall training. Experiments on MNIST were conducted using the Nebula platform, exploring different delay durations, topologies, and attacker proportions. **Table 12** summarizes the configurations evaluated.

Unlike poisoning, this attack targets efficiency. Fig. 7 presents the total training time by node type, showing how lag propagates and increases desynchronization when no mitigation is applied. For reference, the baseline training duration without an attack was 5 minutes; with reputation enabled, it increased slightly to ~6 minutes. To assess how reputation adapts, Fig. 5 shows average reputation evolution under different attack timings. Figures (a) and (b) correspond to the global reputation for attacks starting in round 7 and round 1, respectively. Figures (c) and (d) show the reputation of malicious nodes in the same scenarios. Finally, (e) and (f) present global and malicious reputations under intermittent attacks with intervals of 2, 3, and 4.

Fig. 6 shows round gaps between benign and malicious nodes. Subfigure (a) refers to scenarios where the attack begins in round 1; (b) shows scenarios with the attack starting from round 7. In both cases, the average reputation of malicious nodes is included for correlation.

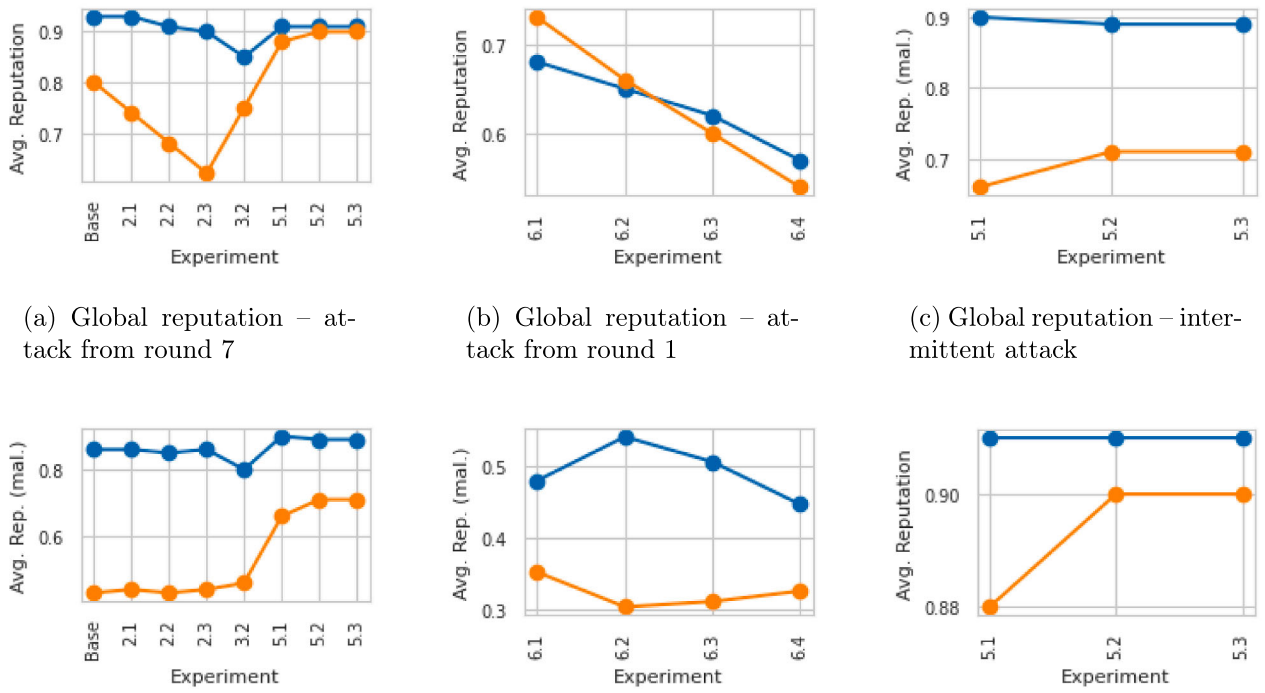
Lastly, Fig. 8 reports the average number of aggregated models depending on the percentage of malicious nodes. Panel (a) corresponds to round 9 (attack starting at round 7), while panel (b) corresponds to round 2 (attack starting at round 1). For example, when 30% of the nodes are malicious in a federation of 10 participants, the average number of aggregated models is around 7. This means that RepuNet successfully identifies and excludes the three low-reputation nodes from the aggregation process. Although these nodes remain in the federation and can continue exchanging information, their model updates are not included in the global aggregation until their reputation recovers. These results demonstrate how RepuNet mitigates delay and poisoning attacks by preserving contributions from trustworthy nodes while isolating unreliable participants.

5.4. Flooding attack

The *flooding* attack aims to saturate the communication layer by overwhelming the network with a high volume of messages from malicious nodes. This behavior degrades communication efficiency, increases CPU overhead, and can ultimately impact model convergence. RepuNet addresses this threat by penalizing nodes that exhibit abnormal message volume. Table 13 summarizes the configurations evaluated for this attack, including variations in topology, proportion of malicious nodes, activation round, and message injection interval.

Fig. 9 shows the evolution of the average reputation under flooding scenarios starting at round 7. Malicious nodes exhibit a sharp, consistent decline in reputation, while benign participants maintain high reputations, highlighting RepuNet's ability to penalize anomalies without affecting honest contributions.

Additional scenarios were tested in which the flooding attack was activated in the first round and ran through round 10. In these early-



(a) Global reputation – attack from round 7

(b) Global reputation – attack from round 1

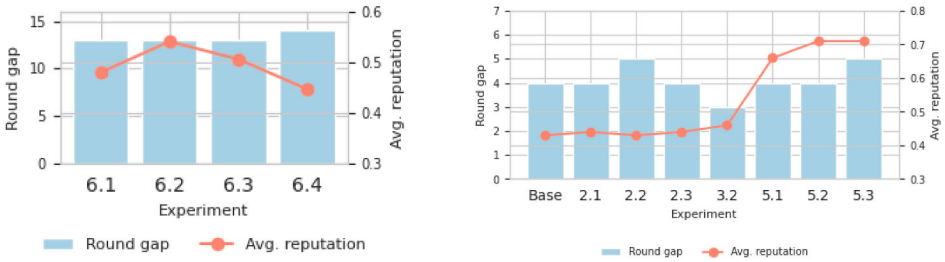
(c) Global reputation – intermittent attack

(d) Malicious reputation – attack from round 7

(e) Malicious reputation – attack from round 1

(f) Malicious reputation – intermittent attack

Fig. 5. Evolution of average reputation across different *delay* attack configurations. Each row compares global reputation (left) and malicious node reputation (right) under a specific attack timing: from round 7, from round 1, and with intermittent activation.



(a) Round 2 – attack from round 1

(b) Round 9 – attack from round 7

Fig. 6. Relationship between the round gap and average reputation of malicious nodes in two scenarios.

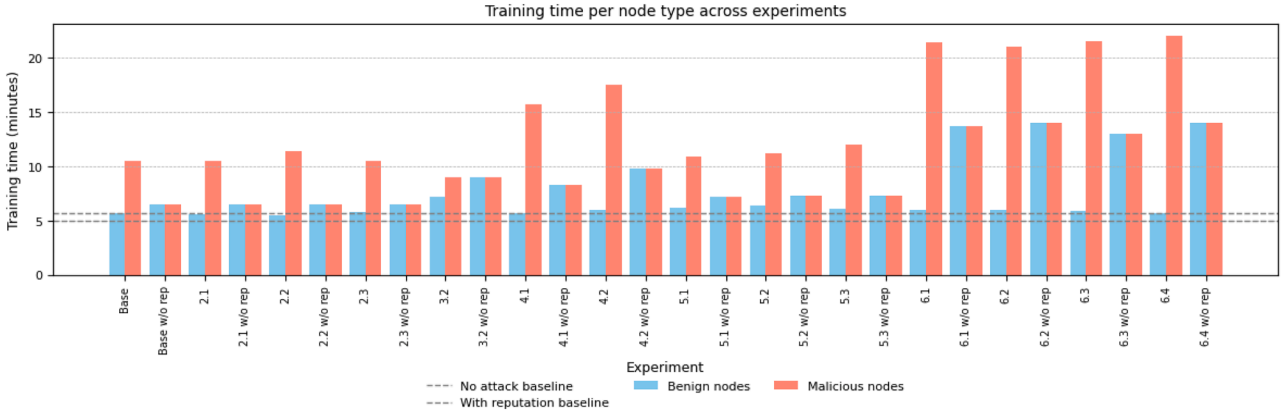
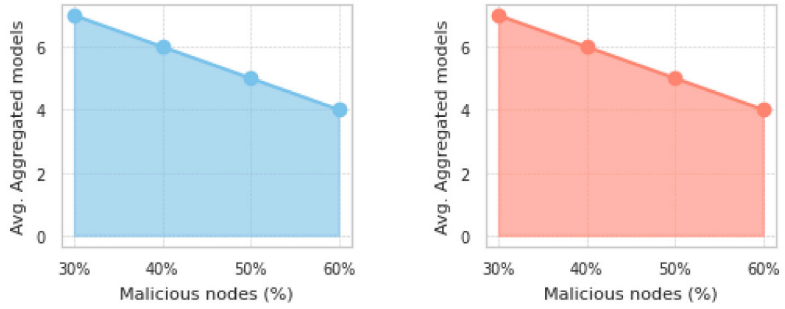


Fig. 7. Total training time per node type in the different scenarios.

Table 10
Performance under structured model poisoning attacks (B = before, D = during, A = after).

Type of Attack	Attack	Reputation	F1 (B)	F1 (D)	F1 (A)	Mean Rep. (B)	Mean Rep. (D)	Mean Rep. (A)
Clean Baseline (FedAvg)	No	No	0.7207	0.9503	0.9622	–	–	–
GLL Neuro Inversion	Yes	No	0.7359	0.2423	0.0203	–	–	–
GLL Neuro Inversion	Yes	Yes	0.7392	0.6757	0.7737	0.8186	0.7363	0.7027
Signed Neuron Remapping (SNR)	Yes	No	0.7312	0.7045	0.6986	–	–	–
Signed Neuron Remapping (SNR)	Yes	Yes	0.7123	0.7656	0.8467	0.8826	0.6792	0.6712



(a) Round 9 – attack from round 7

(b) Round 2 – attack from round 1

Fig. 8. Average number of aggregated models as a function of the percentage of malicious nodes. Each point represents the mean number of models effectively aggregated by RepuNet after filtering out delayed or malicious updates. Since each model corresponds to a participating node, the number of aggregated models directly reflects the proportion of benign participants. Panel (a) shows the results at round 9 when the attack starts at round 7, and panel (b) shows round 2 when the attack starts at round 1.

Table 11
Behavioral separation at round 8 under structured poisoning attacks.

Type of Attack	Reputation	F1 GAP	F1 Variance
GLL Neuro Inversion	No	0.1676	0.0059
GLL Neuro Inversion	Yes	0.9237	0.1777
Signed Neuron Remapping (SNR)	No	0.1784	0.0314
Signed Neuron Remapping (SNR)	Yes	0.0611	0.0721

Table 12
Summary of delay attack configurations tested.

Scenario Group	Topology	Nº nodes	Delay (s)	Malicious Nodes	Interval
Base / 2.x	Fully	10	20	30–60%	1
3.2	Random	10	30	30%	1
4.x	Fully	10	40–60	30%	1
5.x (intermittent)	Fully	10	80	30%	2–4
6.x	Fully	10	100	30–60%	1

Table 13
Flooding attack scenarios: configuration summary (grouped).

Scenario Group	Topology	Nº nodes	Malicious Nodes	Start Round
Base / 2.x	Fully	10	30–60%	7
3.1	Random	10	30%	7
4.1	Fully	10	30%	7 (interval 2)
5.x	Fully	10	30–60%	1

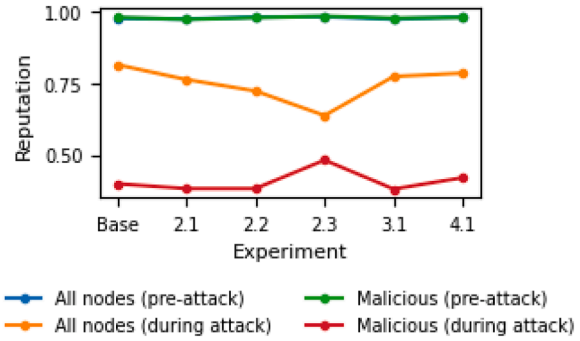


Fig. 9. Average reputation evolution of all and malicious nodes during the flooding attack (activated in round 7).

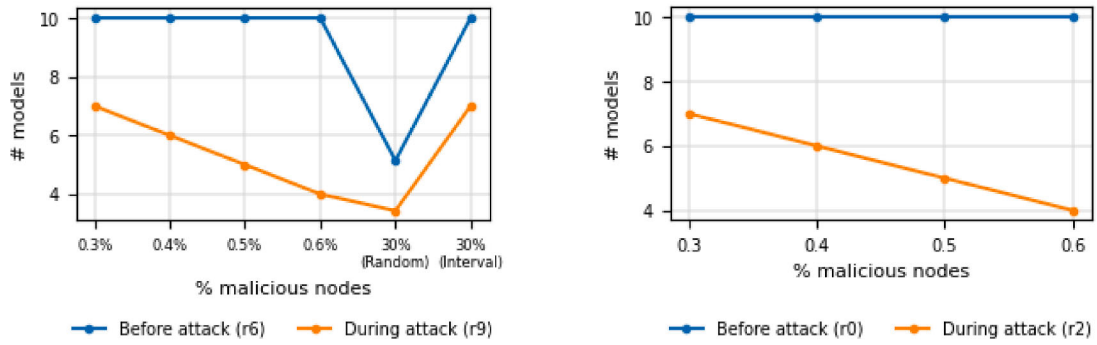
Table 14
Benign and malicious reputation under *flooding* attacks starting at round 1.

Scenario	Benign Rep. (r2)	Benign Rep. (r6)	Malicious Rep. (r2)	Malicious Rep. (r6)
5.1	0.9387	0.9568	0.3256	0.2718
5.2	0.9385	0.9347	0.2936	0.2970
5.3	0.9050	0.9335	0.3025	0.3239
5.4	0.9168	0.9313	0.2950	0.3914

activation cases, round 0 served as the only observation phase, with all nodes initialized to a default reputation of 0.6. Table 14 presents the reputation scores observed in these scenarios.

The impact on the aggregation process was also assessed by measuring the average number of models accepted in round 9. As shown in Fig. 10, the number of aggregated models decreases proportionally with the increase in malicious nodes. This behavior holds for both early and delayed attack activation, confirming RepuNet's robustness even with limited prior information.

Finally, to evaluate the indirect cost of flooding attacks, Table 15 reports the variation in average CPU usage observed in benign nodes. This variation reflects the computational overhead caused by processing excessive messages. Scenarios with higher percentages of malicious nodes tend to exhibit increased CPU load, underscoring the importance of mitigating such attacks not only to preserve aggregation quality but also to reduce resource consumption.



(a) Attack from round 7 (b) Attack from round 1

Fig. 10. Average number of models aggregated in round 9 under different flooding attack activation points. The Y-axis represents the average number of models aggregated by the federation nodes. The system consistently reduces the inclusion of malicious contributions as their proportion increases.

Table 15
CPU usage variation in benign nodes under *flooding* attacks.

Scenario	Attack round	Avg. CPU Variation (%)
Base	7–12	5.23
2.1	7–12	-1.03
2.2	7–12	15.14
2.3	7–12	3.70
3.1	7–12	11.21
4.1	7–12	-4.49
5.1	1–10	-8.14
5.2	1–10	5.70
5.3	1–10	-4.80
5.4	1–10	-9.90

Table 16
Impact of exclusion threshold on performance and aggregation behavior.

Scenario	F1-score (r8)	Avg. Aggregated models (r8)
0.5	0.6238	9
0.6	0.6800	7
0.7	0.6660	7

5.5. Ablation study on the exclusion threshold

To analyze the sensitivity of RepuNet to the exclusion threshold parameter, we conducted an ablation study under a representative model poisoning scenario (Base configuration). The exclusion threshold determines whether a neighbor's model is incorporated into aggregation or discarded when its reputation falls below a predefined value.

The experimental setup corresponds to the Base configuration described in 5.2 (MNIST dataset, 10 fully connected nodes, 30% malicious participants, Dirichlet $\alpha = 0.5$, and attack active from rounds 7 to 10), ensuring that only the exclusion threshold varies across experiments.

We evaluated three threshold values: 0.5, 0.6 (default configuration), and 0.7. For each configuration, we measured (i) the F1-score at round 8 (during the attack window) and (ii) the number of aggregated models at round 8, reflecting the system's filtering strictness under adversarial conditions. Table 16 summarizes the results.

The results indicate a clear impact of the exclusion threshold on robustness and aggregation behavior. A lower threshold (0.5) allows a higher number of models to be aggregated (9 out of 10), which increases participation but reduces robustness under adversarial conditions, resulting in a lower F1-score (0.6238). Conversely, thresholds of 0.6 and 0.7 lead to stricter filtering, limiting aggregation to 7 models and improving attack mitigation. Although 0.7 slightly reduces performance compared to the default configuration, the results confirm that 0.6 provides the best balance between robustness and collaboration in this scenario. Overall, RepuNet demonstrates stable behavior across a reasonable threshold range without requiring fine-grained tuning.

6. Discussion

The experimental evaluation demonstrates that RepuNet provides effective protection against diverse adversarial behaviors in DFL scenarios. Rather than reiterating the internal mechanics of the reputation engine, this section analyzes its observed behavior, highlighting its strengths, limitations, and areas for improvement. Across all experiments, RepuNet consistently reduced the influence of malicious nodes during aggregation without compromising the participation structure of the network. Reputation trajectories showed rapid penalization after attack activation, confirming the system's responsiveness. However, certain patterns, such as intermittent or oscillating attacks, revealed scenarios in which nodes could recover influence too quickly, suggesting that stricter memory-based penalization strategies may be needed. The ability to adapt dynamically to contextual changes, without centralized control or static heuristics, positioned RepuNet as a flexible defense strategy. Nonetheless, further refinement of metric weighting and exclusion thresholds could enhance robustness under more stealthy or coordinated threats. The following subsections delve into specific attack types and their mitigation outcomes.

6.1. Model poisoning

Table 6 presents the F1-scores obtained with and without RepuNet across several poisoning scenarios. The most significant improvements were observed in scenarios *Base*, *3.2*, and *6.1*, where RepuNet increased performance by over 30 points in F1-score, reaching differences above 0.35. In early-activation cases such as *5.3*, where attacks began at round 1, the gap was even greater: F1-score improved from 0.0720 to 0.5166. This demonstrates RepuNet's capacity to recover learning even under high pressure and without a prior observation phase. Scenarios with skewed data distributions (e.g., *5.2* with $\alpha = 0.1$) showed lower gains, as expected under higher heterogeneity, but still maintained improvements over the baseline. In intermittent scenarios (*7.1*, *7.2*), RepuNet showed limited effect, suggesting the system could benefit from memory-based metrics to counter low-frequency adversarial patterns. Overall, the results confirm that RepuNet substantially mitigates poisoning attacks across various conditions, adapting to both topological and distributional challenges.

6.1.1. Baseline robust aggregation methods under model poisoning

Under model poisoning, Krum without attack achieves an F1-score of 0.1764 in the baseline configuration. When poisoning is activated without reputation, the F1-score decreases during the attack phase (0.2152),

but does not collapse, confirming that Krum already provides a degree of geometric robustness against anomalous updates. After the attack interval, performance partially recovers (0.3089), indicating that the aggregation rule limits catastrophic degradation but does not fully isolate adversarial influence.

When reputation is incorporated, the aggregate F1-score does not exhibit a dramatic increase (e.g., 0.2162 during the attack phase and 0.2707 after the attack). This suggests that predictive performance alone does not fully capture the contribution of the reputation mechanism in this setting, as Krum already filters extreme updates at the geometric level.

However, structural analysis at round 8 reveals a more relevant effect. Without reputation, Krum produces a noticeable performance gap between benign and malicious nodes (GAP = 0.2700; variance = 0.0124), reflecting partial behavioral differentiation. Enabling reputation slightly increases this separation (GAP = 0.2845) and amplifies the global variance (0.0196), reinforcing the distinction between trustworthy and adversarial participants. While the aggregate F1-score remains similar, reputation strengthens dynamic trust modeling and improves structural interpretability during adversarial phases.

The impact of reputation is more pronounced under Trimmed Mean aggregation. Without defense, poisoning causes global contamination during the attack phase (F1(D) = 0.1493), with limited recovery afterward (0.3277). Structural separation is minimal (GAP = 0.0557; variance = 0.00067), indicating that benign and malicious contributions remain insufficiently differentiated.

When RepuNet is enabled, the system exhibits improved robustness during the attack (F1(D) = 0.2623) and enhanced post-attack recovery (0.4046). More importantly, structural separation becomes clearly visible (GAP = 0.3106; variance = 0.0250), effectively isolating adversarial behavior. Although the increase in aggregate F1-score is moderate, the substantial rise in behavioral differentiation demonstrates that reputation prevents systemic degradation by attenuating malicious influence while preserving benign contributions.

Overall, these results indicate that in robust aggregation settings the primary contribution of RepuNet is not necessarily a large increase in aggregate F1-score, but rather the reinforcement of behavioral separation, dynamic trust differentiation, and structural robustness across attack phases. Reputation operates as a complementary layer that stabilizes aggregation dynamics beyond the static filtering mechanisms of Krum and Trimmed Mean.

While the previous analysis demonstrates that RepuNet acts as a complementary stabilization layer when combined with robust aggregation rules such as Krum and Trimmed Mean, these mechanisms are primarily designed to filter extreme or geometrically distant updates. More sophisticated attacks may preserve global magnitude properties while introducing coordinated directional bias, potentially bypassing static robustness criteria. The following subsection evaluates whether reputation-based behavioral differentiation remains effective under these stronger and structured poisoning strategies.

6.1.2. Stronger and structured poisoning attacks

Table 10 reports the F1-scores obtained under structured poisoning strategies across the three evaluation phases: before (B), during (D), and after (A) the attack. Unlike simple perturbation-based attacks, both GLL Neuro Inversion and Signed Neuron Remapping (SNR) preserve global parameter magnitude while introducing coordinated directional distortions, challenging aggregation mechanisms that rely primarily on norm or distance filtering.

Under GLL Neuro Inversion, standard FedAvg experiences severe degradation during the adversarial phase, with F1 decreasing from 0.7359 (B) to 0.2423 (D), and collapsing to 0.0203 (A) after the attack interval. This behavior indicates that coordinated directional bias can destabilize aggregation even when global magnitude consistency is preserved, preventing post-attack recovery of the global model.

When RepuNet is enabled, degradation during the attack phase is significantly mitigated (F1(D) = 0.6757), and post-attack convergence is restored (F1(A) = 0.7737). The progressive decrease in mean reputation values (0.8186 → 0.7363 → 0.7027) reflects sustained penalization of structurally inconsistent participants. Structural analysis at round 8 further confirms this differentiation: without reputation, separation remains limited (GAP = 0.1676; variance = 0.0059), whereas enabling reputation produces strong behavioral isolation (GAP = 0.9237; variance = 0.1777). The increase in variance reflects deliberate structural separation rather than instability, as benign nodes maintain high performance while malicious nodes collapse. This result confirms that the distribution-aware similarity clustering mechanism introduced in Eqs. 25 y 28 are critical for identifying magnitude-preserving adversarial coordination, which cannot be detected through norm-based filtering alone.

Under Signed Neuron Remapping (SNR), the attack exhibits a more subtle impact. Without reputation, F1 during the attack reaches 0.7045 and, after the adversarial interval, only slightly recovers to 0.6986. This indicates that the accumulated degradation during the attack window continues to affect the learning trajectory, limiting the model's recovery capacity.

With RepuNet enabled, improvement is observed both during and after the attack. During the adversarial phase, F1 increases from 0.7045 to 0.7656, confirming that the reputation mechanism reduces the structural impact of malicious updates even when 30% of the nodes remain adversarial. Although the global mean is partially influenced by malicious participants during the attack rounds, benign nodes continue refining the model without experiencing critical degradation.

The most significant difference appears in the post-attack phase. Without reputation, F1 after the attack stabilizes at 0.6986, whereas with reputation it reaches 0.8467, representing an improvement of more than 0.15 points. This substantial gain demonstrates that the mechanism not only mitigates immediate degradation during adversarial rounds but also protects the long-term convergence trajectory. During the attack window, reputation progressively attenuates the influence of malicious nodes; once the attack ends, these participants no longer significantly affect aggregation due to their accumulated low reputation, allowing benign nodes to continue refining the global model toward stable and significantly higher convergence.

From a structural perspective, the pattern under SNR differs from that observed in GLL. Although variance increases moderately (0.0314 → 0.0721), the GAP decreases (0.1784 → 0.0611), indicating that the reputation mechanism induces gradual influence redistribution rather than abrupt behavioral collapse. In contrast to GLL, where reputation generates strong polarization between benign and malicious nodes, under SNR it acts as a progressive regulatory mechanism that reduces adversarial influence without inducing extreme separation. The final outcome is not radical fragmentation, but a clear and sustained improvement in global performance. This behavior suggests that under magnitude-preserving structured attacks such as SNR, reputation operates as a gradual attenuation mechanism rather than a binary isolation filter, progressively reducing adversarial influence while preserving overall convergence stability.

Overall, structured poisoning experiments confirm that RepuNet mitigates degradation during adversarial rounds, preserves post-attack recovery by preventing low-reputation nodes from distorting aggregation dynamics, and reinforces robustness against coordinated magnitude-preserving attacks that manipulate directional alignment rather than parameter norms.

6.2. Delay attack

As shown in Table 12, the delay attack was tested under various delays (20-100 seconds), attacker proportions, and topologies. RepuNet successfully penalized delayed nodes across all configurations. In early rounds (r2), scenarios such as *2.3* and *6.1* already showed reputation

gaps of more than 0.3 between benign and malicious participants. Reputation remained low for attackers in both fully connected and random topologies, as shown in Fig. 5, confirming the system's capacity to suppress desynchronizing behavior. Intermittent delay scenarios revealed partial reintegration between inactive phases, which may be addressed by increasing the exclusion threshold or applying temporal penalties. Despite this, overall training time remained stable for honest nodes, while malicious participants completed fewer rounds (Fig. 7). The reduction in model aggregation under delay conditions was also consistent. Fig. 8 shows that as the delay severity or attacker proportion increased, the number of models selected decreased accordingly, particularly in scenarios *4.1* and *6.3*.

6.3. Flooding attack

Table 13 summarizes the configurations evaluated, including topology, proportion of malicious nodes, activation round, and interval. When the flooding attack was activated at round 7 (scenarios *Base* to *4.1*), RepuNet responded effectively: the reputation of malicious nodes dropped from initial values near 0.98 to values between 0.38 and 0.48. In contrast, honest nodes maintained values around 0.80. In early-activation scenarios (round 1, *5.1*–*5.4*), all nodes started from a neutral reputation of 0.6. Table 14 shows the evolution of reputation for benign and malicious nodes. Benign nodes rapidly recovered to values above 0.93 by round 6, while adversarial nodes remained low or exhibited only limited recovery. The number of aggregated models in round 9 (Fig. 10) decreased progressively with the proportion of malicious nodes, both for early and delayed attacks, indicating consistent exclusion of unreliable contributions. Intermittent attacks such as *4.1* revealed partial reintegration of malicious nodes between flooding bursts, suggesting a limitation of short-term metrics. Despite this, RepuNet re-penalized such behavior in subsequent rounds, preserving overall robustness. Table 15 presents CPU usage variations for benign nodes. In most cases, RepuNet reduced or stabilized overhead, with the highest efficiency gains observed in scenarios *5.1* and *5.3*. These results confirm that reputation-based filtering improves not only model quality but also resource efficiency under flooding conditions.

6.4. Impact of reputation feedback on communication overhead

To isolate the communication overhead introduced by the reputation feedback mechanism, we conducted experiments without adversarial behavior. In the MNIST setting, the baseline training time without reputation was approximately 5.0 minutes, while enabling reputation feedback increased it to around ∼ 6 minutes. This overhead corresponds to the additional exchange of reputation-related messages per round. Importantly, CPU utilization remained stable and convergence behavior was unaffected, indicating that the communication overhead introduced by RepuNet is limited and does not significantly impact overall system efficiency.

7. Conclusions

This work presented a reputation mechanism for DFL that detects and mitigates malicious behavior during training. The system evaluates each node based on locally observable metrics, such as model similarity, parameter changes, latency, and message volume, and dynamically updates reputation scores. Nodes with low reputation are excluded from aggregation, while recovery is possible upon improved behavior. Experiments confirm the system's effectiveness against model poisoning, delay, and flooding attacks. In all cases, malicious nodes suffered a sharp drop in reputation after initiating the attack, enabling their exclusion without degrading the aggregated model.

RepuNet could be extended to address threats such as data poisoning, opening the door to integrating new metrics that assess model coherence with local data. Future improvements could also include a dynamic exclusion threshold rather than a fixed one, and improved weight assignment by analyzing Z-score normalization, Bayesian schemes, or attention mechanisms to adjust metric relevance based on context.

CRediT authorship contribution statement

Isaac Marroquí Penalva: Visualization, Validation, Software, Investigation; **Enrique Tomás Martínez Beltrán:** Visualization, Validation, Supervision, Project administration, Methodology, Funding acquisition, Formal analysis; **Manuel Gil Pérez:** Validation, Supervision, Project administration, Methodology; **Alberto Huertas Celdrán:** Validation, Supervision, Project administration, Methodology.

Data availability

No data was used for the research described in the article.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgment

This work was supported by (a) the Swiss Federal Office for Defense Procurement (armasuisse) with the DECIMAL project (CYD-C-2020003), (b) 21629/FPI/21, Fundación Séneca, Región de Murcia (Spain), (c) the strategic project DEFENDER from the Spanish National Institute of Cybersecurity (INCIBE), by the Recovery, Transformation and Resilience Plan, Next Generation EU, and (d) the European Commission through the Horizon Europe/JU SNS project ROBUST-6G (Grant Agreement no. 101139068).

References

[1] B. McMahan, E. Moore, D. Ramage, S. Hampson, B.A. y Arcas, Communication-efficient learning of deep networks from decentralized data, in: Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, 2017, pp. 1273–1282.

[2] Z. Song, H. Sun, H.H. Yang, X. Wang, Y. Zhang, T.Q.S. Quek, Reputation-based federated learning for secure wireless networks, IEEE Internet Things J. 9 (2) (2022) 1212–1226. https://doi.org/10.1109/JIOT.2021.3079104

[3] J. Kang, Z. Xiong, D. Niyato, S. Xie, J. Zhang, Incentive mechanism for reliable federated learning: a joint optimization approach to combining reputation and contract theory, IEEE Internet Things J. 6 (6) (2019) 10700–10714. https://doi.org/10.1109/JIOT.2019.2940820

[4] Y. Zhao, J. Zhao, L. Jiang, R. Tan, D. Niyato, Z. Li, L. Lyu, Y. Liu, Privacy-Preserving Blockchain-Based Federated Learning for IoT Devices, 2021, https://doi.org/10.48550/arXiv.1906.10893. arXiv:2303.16668

[5] J. Domingo-Ferrer, A. Blanco-Justicia, J. Manjón, D. Sánchez, Secure and privacy-preserving federated learning via co-utility, IEEE Internet Things J. 9 (5) (2021) 3988–4000.

[6] M. Panigrahi, S. Bharti, A. Sharma, A reputation-aware hierarchical aggregation framework for federated learning, Comput. Electr. Eng. 111 (2023) 108900.

[7] L. Gao, L. Li, Y. Chen, C. Xu, M. Xu, FGFL: a blockchain-based fair incentive governor for federated learning, J. Parallel Distrib. Comput. 163 (2022) 283–299.

[8] T. Nguyen, P. Rieger, M. Miettinen, A.-R. Sadeghi, Poisoning attacks on federated learning-based IoT intrusion detection system, in: Workshop on Decentralized IoT Systems and Security, 2020, pp. 1–7. https://doi.org/10.14722/diss.2020.23003

[9] Y. Li, X. Wei, Y. Li, Z. Dong, M. Shahidehpour, Detection of false data injection attacks in smart grid: a secure federated deep learning approach, IEEE Trans. Smart Grid 13 (6) (2022) 4862–4872. https://doi.org/10.1109/TSG.2022.3204796

[10] J. Li, L. Lyu, X. Liu, X. Zhang, X. Lyu, FLEAM: A federated learning empowered architecture to mitigate DDos in industrial IoT, IEEE Trans. Ind. Inf. 18 (6) (2022) 4059–4068. https://doi.org/10.1109/TII.2021.3088938

[11] M. Zang, C. Zheng, T. Koziak, N. Zilberman, L. Dittmann, Federated learning-based in-network traffic analysis on IoT edge, in: 2023 IFIP Networking Conference (IFIP Networking), 2023, pp. 1–6. https://doi.org/10.23919/IFIPNetworking57963.2023.10186438

[12] F.P.-C. Lin, S. Hosseinalipour, N. Michelusi, C.G. Brinton, Delay-aware hierarchical federated learning, IEEE Trans. Cognit. Commun. Netw. 10 (2) (2024) 674–688. https://doi.org/10.1109/TCCN.2023.3329024

[13] E. Hallaji, R. Razavi-Far, M. Saif, Q. Yang, Decentralized federated learning: a survey on security and privacy, IEEE Trans. Big Data 10 (2024) 194–213. https://doi.org/10.1109/TBDATA.2024.3362191

[14] S. Yuan, B. Cao, Y. Sun, Z. Wan, M. Peng, Secure and efficient federated learning through layering and sharding blockchain, IEEE Trans. Network Sci. Eng. 11 (3) (2024) 3120–3134. https://doi.org/10.1109/TNSE.2024.3361458

[15] H. Wang, H. Zhang, L. Wang, S. Xuan, Q. Zhang, Fedeval: defending against lazybone attack via multi-dimension evaluation in federated learning, ACM Trans. Sens. Netw. 21 (1) (2025) 1–23. https://doi.org/10.1145/3703631

[16] CyberDataLab, Nebula: A Platform for Decentralized Federated Learning, 2025, (https://github.com/CyberDataLab/nebula). Accessed: 2025-06-13.

[17] M. Fang, Z. Zhang, Hairi, P. Khanduri, J. Liu, S. Lu, Y. Liu, N. Gong, Byzantine-robust decentralized federated learning, in: Proceedings of the 2024 on ACM SIGSAC Conference on Computer and Communications Security, CCS '24, Association for Computing Machinery, New York, NY, USA, 2024, p. 2874-2888. https://doi.org/10.1145/3658644.3670307

[18] S. Guo, T. Zhang, H. Yu, X. Xie, L. Ma, T. Xiang, Y. Liu, Byzantine-resilient decentralized stochastic gradient descent, IEEE Trans. Circuits Syst. Video Technol. 32 (6) (2022) 4096–4106. https://doi.org/10.1109/TCSVT.2021.3116976

[19] M. Fang, X. Cao, J. Jia, N. Gong, Local model poisoning attacks to byzantine-robust federated learning, in: 29th USENIX Security Symposium (USENIX Security 20), USENIX Association, 2020, pp. 1605–1622. https://www.usenix.org/conference/usenixsecurity20/presentation/fang.

[20] G. Baruch, M. Baruch, Y. Goldberg, A little is enough: circumventing defenses for distributed learning, in: H. Wallach, H. Larochelle, A. Beygelzimer, F.d. Alché-Buc, E. Fox, R. Garnett (Eds.), Advances in Neural Information Processing Systems, 32, Curran Associates, Inc., 2019. https://proceedings.neurips.cc/paper_files/paper/2019/file/ec1c59141046cd1866bbbcdfb6ae31d4-Paper.pdf.