Game-Theoretic Stabilization of Dynamic IoT Federated Learning via Agent-Guided Stackelberg Policies

author names withheld

Under Review for NExT-Game 2026

Abstract

Dynamic Internet-of-Things (IoT) federated learning (FL) operates under incomplete client information, where devices frequently join or leave, local data are heterogeneous, and some participants may be unreliable or malicious. Existing robust FL defenses often rely on static aggregation rules or fixed thresholds, which may be insufficient under time-varying system conditions. In this paper, we propose Trust-aware Stackelberg Federated Learning (TS-FL), a Stackelberg-based policy stabilization framework for dynamic IoT FL under zero-trust principles. The server acts as a Stackelberg leader, while clients act as followers responding to rewards and participation costs. To infer hidden client reliability, the server extracts information-theoretic evidence from model-update differences using entropy, KL-divergence, and mutual information, and converts it into robust anomaly scores through MAD-based normalization. These scores are accumulated using a Bayesian trust ledger to distinguish temporary abnormality from repeated suspicious behavior. An LLM agent serves as a strategic orchestrator, but outputs only a bounded threat level rather than directly assigning aggregation weights, trust scores, or exclusion decisions. This threat level is mapped to policy parameters through fixed formula-driven rules, improving transparency and reproducibility. TS-FL jointly regulates trust-aware aggregation, reward allocation, and client exclusion, thereby stabilizing FL in dynamic IoT environments with incomplete client information.

1. Introduction

1.1. Background and Motivation

Federated learning (FL) is well-suited for sensitive IoT data, but dynamic participation, non-IID distributions, and malicious clients make implicit trust unrealistic [6, 7]. Although Zero Trust principles require continuous verification, existing robust FL defenses often rely on static aggregation rules or fixed thresholds, limiting their ability to adapt to evolving IoT environments [8].

We propose Trust-aware Stackelberg Federated Learning (TS-FL), a Stackelberg-based policy stabilization framework for dynamic IoT FL under incomplete information. The server acts as a leader and stabilizes aggregation by combining information-theoretic evidence from client updates with Bayesian trust accumulation. Specifically, entropy, KL-divergence, and mutual information are extracted from model-update differences and accumulated over rounds to distinguish transient noise from persistent malicious behavior. A strategic LLM agent observes recent system dynamics and outputs only a bounded threat level, rather than directly assigning aggregation weights or exclusion decisions. This threat level is mapped to trust thresholds, reward decay, and other Stackelberg policy parameters through fixed deterministic formulas, preserving numerical grounding, transparency, and reproducibility.

© .

Overall, this paper contributes TS-FL, a game-theoretic trust stabilization framework that integrates information-theoretic client profiling, Bayesian trust memory, and formula-grounded LLM orchestration for robust FL in dynamic IoT environments.

1.2. Related Work

Prior secure FL research covers robust aggregation [1, 2, 8], trust-aware FL [3], information-theoretic FL [4, 10], game-theoretic FL [5, 12], and LLM-based control [9]. However, few jointly address dynamic participation, incomplete information, and numerically grounded LLM orchestration in IoT environments. Although agent-based client selection has been explored for FL efficiency [11], it does not provide formula-constrained security policy control. We bridge this gap by formulating dynamic IoT FL as an incomplete-information Stackelberg policy stabilization problem; a detailed positioning is provided in Table 3 (Appendix A).

2. Proposed Framework

2.1. Overall Architecture

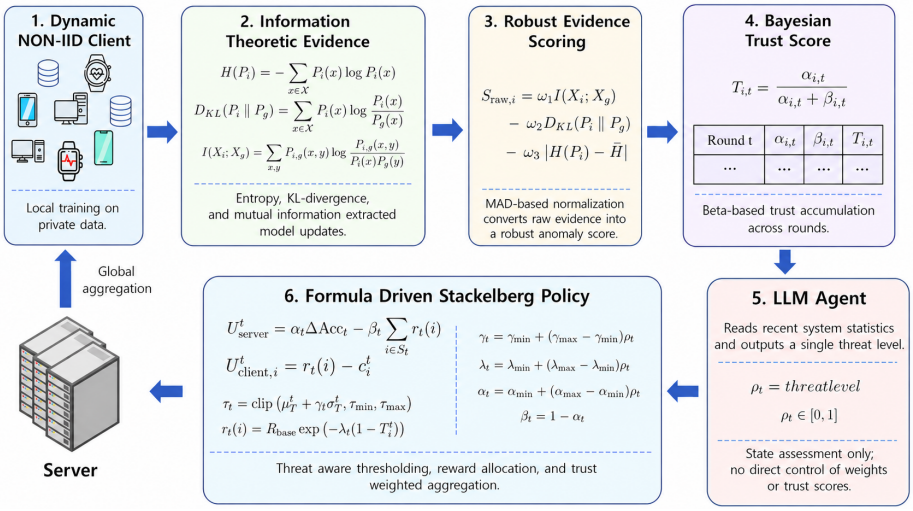

Figure 1: Framework architecture: IT evidence and Bayesian trust profile clients, while an LLM-derived threat level orchestrates Stackelberg policy for adaptive aggregation and exclusion.

Figure 1 illustrates the proposed framework. We consider a dynamic IoT FL environment where the participating set S_t fluctuates due to churn and potential adversarial behavior, forcing the server to operate under incomplete information. To implement zero-trust security, the framework processes updates through four main stages before aggregation:

First, rather than direct aggregation, the server extracts **information-theoretic evidence** (entropy, KL-divergence, and mutual information) from model-update differences to profile client behavior. Second, these indicators are passed to a **Bayesian trust module**, which maintains a persistent ledger to distinguish temporary noise from repeated malicious activity through strike-based exclusion and trust decay for non-participants. Third, an **LLM agent** serves as a strategic Stackelberg orchestrator. It observes system-level dynamics and outputs a bounded threat level $\rho_t \in [0, 1]$, which is then

mapped to deterministic policy parameters through fixed formulas to ensure reproducibility and numerical grounding. Finally, the server combines the trust scores and policy parameters to determine weights, rewards, and **pre-aggregation exclusion** decisions, ensuring that suspicious updates at round t are filtered before they can affect the global model.

2.2. Information-Theoretic Evidence and Bayesian Trust

For each client i at round t, the server computes the update difference $\Delta W_{i,t} = \mathrm{flat}(W_{i,t}) - \mathrm{flat}(W_{g,t})$ and its empirical distribution P_i. To profile behavior, we extract an evidence vector $\mathbf{e}_{i,t} = [H(P_i), D_{KL}(P_i \parallel P_g), I(X_i; X_g)]$, where the components denote entropy, KL-divergence, and mutual information, respectively. These indicators are combined into a scalar anomaly score:

$$S_{\mathrm{raw},i} = \omega_1 I(X_i; X_g) - \omega_2 D_{KL}(P_i \parallel P_g) - \omega_3 \left| H(P_i) - \bar{H} \right|, \tag{1}$$

where $\bar{H}$ is the round-wise mean entropy. This score rewards updates aligned with the global model while penalizing distributional deviations.

To ensure robustness, raw scores are normalized via Median Absolute Deviation (MAD) to obtain an evidence score z_i, which updates a Bayesian trust ledger. Client reliability is modeled via a Beta distribution:

$$T_{i,t} = \mathbb{E}[\mathrm{Beta}(\alpha_{i,t}, \beta_{i,t})] = \frac{\alpha_{i,t}}{\alpha_{i,t} + \beta_{i,t}}. \tag{2}$$

Parameters α_i and β_i accumulate behavioral evidence over rounds, enabling the system to distinguish transient noise from persistent attacks. Dynamic participation is handled through trust regression for non-participants and strike-based exclusion for repeatedly suspicious clients (see Appendix B for detailed update rules).

2.3. LLM-Agent-Guided Stackelberg Policy Orchestration

We formulate the interaction between the FL server and clients as a Stackelberg game, where the server acts as the leader and clients act as followers. The server aims to stabilize global learning by adjusting policy parameters, while clients decide whether to participate based on their expected rewards and costs. Unlike classical Stackelberg formulations that assume static payoffs and complete information, our setting involves time-varying client populations, partially observed reliability, and non-stationary attack behaviors. Under such conditions, deriving a closed-form Stackelberg equilibrium is generally intractable, since both the leader's policy parameters and the followers' best responses depend on evolving trust states and unobserved adversarial strategies.

Therefore, we adopt a policy-stabilization perspective rather than explicit equilibrium computation. At each communication round, the server commits to a bounded policy mapping, and clients respond myopically according to their current expected utility. This round-wise commitment structure preserves the leader–follower hierarchy of Stackelberg interaction while remaining tractable under incomplete information. In this framework, the LLM agent serves as a high-level threat assessor that observes recent system-level statistics, such as model performance, trust distribution, and client churn, and outputs a bounded threat level $\rho_t \in [0, 1]$. Crucially, the agent does not directly control aggregation weights, trust scores, rewards, or exclusion decisions. Instead, ρ_t is converted into policy parameters through deterministic formula-driven mappings, ensuring reproducibility and numerical grounding.

As ρ_t increases, the framework becomes more conservative by increasing the trust-threshold sensitivity and strengthening reward decay for low-trust clients. Specifically, the trust threshold is computed as

$$\tau_t = \mathrm{clip}(\mu_T^t + \gamma_t \sigma_T^t, \tau_{\min}, \tau_{\max}),$$

where μ_T^t and σ_T^t denote the mean and standard deviation of trust scores among participating clients. The reward assigned to client i is determined by

$$r_t(i) = R_{\text{base}} \exp[-\lambda_t(1 - T_{i,t})],$$

which reduces the incentive for low-trust clients to participate under high-threat conditions. After trust filtering, the server performs trust-aware aggregation over the accepted client set $\mathcal{A}_t$, with aggregation weights proportional to both client trust and local data size:

$$w_i^t = \frac{T_{i,t}\sqrt{n_i}}{\sum_{j \in \mathcal{A}_t} T_{j,t}\sqrt{n_j}}.$$

This design ensures that suspicious updates are filtered before aggregation, while reliable and data-rich clients receive proportionally larger influence.

Detailed utility functions and complete policy mapping rules are provided in Appendix C. Empirical convergence and stabilization behavior under this formulation are examined in Section 3 and Appendix F.

The TS-FL procedure operates in four stages: (1) extraction of information-theoretic evidence from local updates (2) MAD-based normalization and Bayesian trust accumulation (3) mapping of an LLM-guided threat level to Stackelberg policy parameters and (4) trust-weighted aggregation of clients filtered by the dynamic threshold. See Appendix D for the complete algorithm.

3. Experimental Results and Analysis

3.1. Experimental Setup

We evaluate the framework on the X-IIoTID dataset for multi-class IoT intrusion detection. The experiments simulate a dynamic cross-device FL environment with stochastic participation and recurring attack waves across four phases: clean convergence (P1), first attack (P2), dormant (P3), and second attack (P4). We compare our method against FedAvg, Krum, coordinate-wise median, and trimmed mean. See Appendix E for details on preprocessing, configurations, and hyperparameters.

3.2. Adaptive Response under Recurring Coordinated-Shift Attacks

Figure 2 and Table 1 evaluate the framework's capacity to detect attacks, adapt policies, and preserve trust memory during recurring coordinated-shift attacks, which are used as the representative attack scenario in the main analysis. During P1, most methods achieve similar convergence, excluding Krum. However, in P2, conventional baselines (FedAvg, trimmed mean, median) degrade significantly to macro-F1 scores of 0.1625, 0.1770, and 0.0689, respectively, while our method maintains 0.7485. Notably, the proposed method recovers adaptively within the same phase after an initial short-term collapse at the onset of the attack wave.

A key distinction emerges in P4: while baselines degrade again after a temporary P3 recovery (e.g., FedAvg: $0.7383 \rightarrow 0.2408$; trimmed mean: $0.7308 \rightarrow 0.3587$), our method sustains stability,

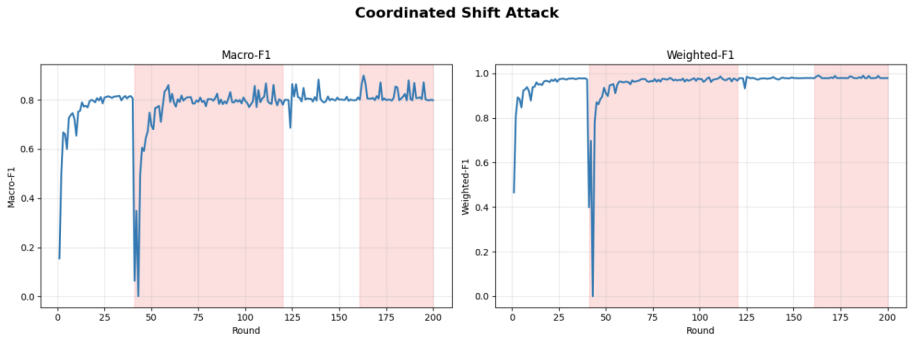

Figure 2: Round-wise behavior of TS-FL under recurring attacks (P1–P4 shaded). The initial dip in P2 confirms attack non-triviality, while stability in P4 demonstrates persistent exclusion of attackers through client-level trust memory.

Table 1: Phase-level performance under recurring coordinated-shift attacks.

Method	**P1 Clean**		**P2 Attack**		**P3 Clean**		**P4 Attack**	
	Acc	F1	Acc	F1	Acc	F1	Acc	F1
Ours	**0.9550**	0.7609	**0.9414**	**0.7485**	**0.9828**	**0.8058**	**0.9855**	**0.8157**
FedAvg	0.9528	0.7586	0.4681	0.1625	0.9408	0.7383	0.5562	0.2408
Krum	0.8758	0.5626	0.6413	0.3233	0.7664	0.3826	0.6690	0.2765
Trimmed Mean	0.9540	0.7621	0.4987	0.1770	0.9329	0.7308	0.6768	0.3587
Median	0.9538	0.7619	0.4187	0.0689	0.9339	0.7059	0.5694	0.2462

even improving from 0.8058 (P3) to 0.8157 (P4). This demonstrates that TS-FL does not merely react to current-round updates but preserves client-level memory of malicious behavior, effectively blocking recurring attackers from regaining influence.

Adaptive trust and policy behavior. Additional analyses in Appendix G show how the LLM-guided policy and Bayesian trust mechanism produce this behavior. When the first coordinated-shift attack begins, the model experiences a sharp macro-F1 collapse, indicating that the attack is non-trivial. The LLM agent subsequently raises the bounded threat level based on system-level evidence, while the deterministic policy mapping increases the conservativeness of the trust threshold and reward decay. At the same time, the Bayesian trust ledger separates benign and malicious clients and hard-drops repeatedly suspicious low-trust clients. As a result, when the same malicious pool is scheduled to reappear in P4, its effective participation is blocked by persistent trust memory rather than rediscovered from scratch.

3.3. Ablation Study

To evaluate each component's contribution, we compare the full model against three variants: Ours w/o Info, Ours w/o Bayes, and Ours w/o LLM. As shown in Table 2, removing Info or Bayesian modules triggers severe macro-F1 collapses in P2 (to 0.1352 and 0.1677) and P4, while the variant without LLM falls short only during initial attacks. This confirms that Info-evidence and Bayesian accumulation ensure fundamental reliability, while LLM orchestration is essential for adaptive responses to sudden attack surges.

Table 2: Ablation results under dynamic attack waves. P1 and P3 denote clean phases, while P2 and P4 denote attack phases.

Method	Overall		P1		P2		P3		P4	
	Acc	F1	Acc	F1	Acc	F1	Acc	F1	Acc	F1
Ours	**0.9612**	**0.7759**	0.9550	0.7609	**0.9414**	**0.7485**	**0.9828**	**0.8058**	**0.9855**	**0.8157**
Ours w/o Info	0.6871	0.3391	0.9567	0.7650	0.3815	0.1352	0.9684	0.7877	0.2141	0.1674
Ours w/o Bayes	0.7221	0.3979	**0.9578**	**0.7672**	0.4321	0.1677	0.9747	0.7952	0.2729	0.1875
Ours w/o LLM	0.9187	0.7046	0.9557	0.7627	0.8467	0.5753	0.9806	0.8050	0.9828	0.8110

3.4. Comparison with Existing FL Defenses

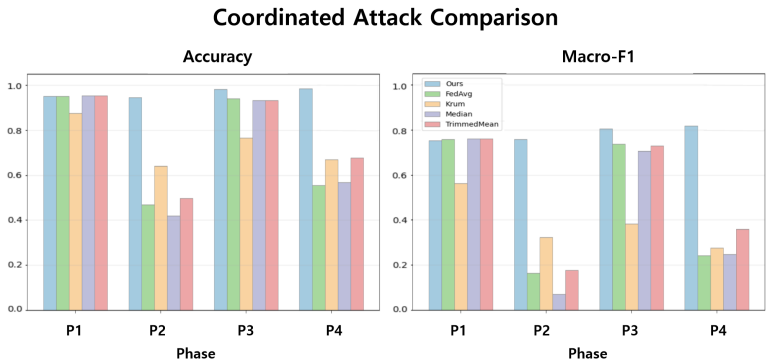

Figure 3: Performance comparison under recurring attacks. Our method remains stable across phases, whereas baselines degrade during attack waves (P2, P4).

We evaluate our method against FedAvg, Krum, median, and trimmed mean under recurring coordinated-shift attacks (P1–P4). As shown in Figure 3, baselines degrade sharply during attack waves (P2, P4) because they lack persistent client-level memory, reacting only to current-round updates. In contrast, the proposed method maintains stability across all phases. By permanently excluding suspicious clients during P2, the framework remains robust when the same malicious pool resurfaces in P4. This demonstrates that persistent trust memory and policy-level stabilization offer superior defense compared to one-shot robust aggregation rules.

4. Conclusion

This paper presented a Stackelberg-based policy stabilization framework for dynamic IoT FL. By integrating information-theoretic evidence, Bayesian trust, and LLM-guided orchestration, our approach enables adaptive defense under zero-trust principles. A key contribution is the use of a bounded LLM threat level mapped to deterministic formulas, ensuring both strategic flexibility and numerical reproducibility. Experiments on the X-IIoTID dataset demonstrate that the framework effectively recovers from initial attacks and, crucially, prevents recurring adversarial influence through persistent client-level trust memory. Compared to conventional robust aggregation rules, our method maintains superior stability across recurring attack waves and diverse adversarial behaviors, including model poisoning and sign-flip attacks. Future work will focus on (i) deriving formal regret bounds and equilibrium characterizations for the proposed online Stackelberg formulation under adversarial client dynamics, and (ii) evaluating the framework's scalability across larger, more diverse IoT datasets and complex participation patterns.

References

[1] Francesco Colosimo and Floriano De Rango. Distance-statistical based byzantine-robust algorithms in federated learning. In *2024 IEEE 21st Consumer Communications & Networking Conference (CCNC)*, pages 1034–1035, 2024.

[2] Mario Garcia-Marquez, Nuria Rodriguez-Barroso, M. Victoria Luzon, and Francisco Herrera. Improving krum's byzantine resilience in federated learning via layerwise aggregation. In *2025 International Joint Conference on Neural Networks (IJCNN)*, 2025.

[3] Jingjing Guo, Zhiquan Liu, Siyi Tian, Feiran Huang, Jiaxing Li, and Xinghua Li. Tfl-dt: A trust evaluation scheme for federated learning in digital twin for mobile networks. *IEEE Journal on Selected Areas in Communications*, 41(11):3548–3560, 2023.

[4] Shayan Mohajer Hamidi, Renhao Tan, Linfeng Ye, and En-Hui Yang. Fed-it: Addressing class imbalance in federated learning through an information- theoretic lens. *2024 IEEE International Symposium on Information Theory*, pages 3548–3560, 2024.

[5] Guangjing Huang, Qiong Wu, Peng Sun, Qian Ma, and Xu Chen. Collaboration in federated learning with differential privacy: A stackelberg game analysis. *IEEE Transactions on Parallel and Distributed Systems*, 35(3):455–469, 2024.

[6] Latif U. Khan, Walid Saad, Zhu Han, Ekram Hossain, and Choong Seon Hong. Federated learning for internet of things: Recent advances, taxonomy, and open challenges. *IEEE Communications Surveys & Tutorials*, 23(3):1759–1799, 2021.

[7] Zili Lu, Heng Pan, Yueyue Dai, Xueming Si, and Yan Zhang. Federated learning with non-iid data: A survey. *IEEE Internet of Things Journal*, 11(11):19188–19209, 2024.

[8] Lingjuan Lyu, Han Yu, Xingjun Ma, Chen Chen, Lichao Sun, and Jun Zhao. Privacy and robustness in federated learning: Attacks and defenses. *IEEE Transactions on Neural Networks and Learning Systems*, 35(7):8726–8746, 2024.

[9] Chamith Mawela, Chaouki Ben Issaid, and Mehdi Bennis. A web-based solution for federated learning with llm-based automation. *IEEE Internet of Things Journal*, 12(12):19488–19503, 2025.

[10] Fu Peng, Meng Zhang, and Ming Tang. An information-theoretic analysis for federated learning under concept drift. *IEEE Transactions on Network Science and Engineering*, 13:1848–1853, 2025.

[11] Sai Qian Zhang, Jieyu Lin, and Qi Zhang. A multi-agent reinforcement learning approach for efficient client selection in federated learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 9091–9099, 2022.

[12] Haitao Zhao, Mengying Zhou, Wenchao Xia, Yiyang Ni, Guan Gui, and Hongbo Zhu. Economic and energy-efficient wireless federated learning based on stackelberg game. *IEEE Transactions on Vehicular Technology*, 73(2):2995–2999, 2024.

Appendix A. Extended Related Work and Positioning

This appendix provides a qualitative positioning of the proposed framework against major research directions in secure and adaptive FL. Robust aggregation mainly addresses Byzantine-resilient update filtering, while trust-aware and information-theoretic FL introduce reliability estimation or statistical evidence. Game-theoretic FL models incentives and participation costs, and recent LLM-agent approaches explore high-level control. However, these directions typically address dynamic participation, incomplete information, trust accumulation, strategic policy adaptation, and LLM grounding in isolation.

Table 3 summarizes this positioning. The proposed framework differs from prior directions by jointly integrating information-theoretic evidence, Bayesian trust accumulation, Stackelberg policy adaptation, and formula-grounded LLM orchestration for dynamic IoT FL.

Table 3: Qualitative positioning against related research directions.

Research line	Dynamic	Incomplete	Trust Accum.	Game Policy	LLM Grounding	Formula Constraint
Robust aggregation FL [1, 2]	△	△	–	–	–	O
Trust-aware FL [3]	△	△	O	–	–	O
Information-theoretic FL [4, 10]	△	△	–	–	–	O
Game-theoretic FL [5, 12]	△	△	–	O	–	O
LLM-agent-based control [9, 11]	O	△	–	–	O	O
Ours	O	O	O	O	O	O

O = explicitly addressed; △ = partially addressed; – = not primarily addressed.

Appendix B. Information-Theoretic Evidence and Bayesian Trust

B.1. Probabilistic Modeling of Weight Updates

For each participating client i at round t, the submitted model-update difference is defined as

$$\Delta W_{i,t} = \mathrm{flat}(W_{i,t}) - \mathrm{flat}(W_{g,t}),$$

where $W_{i,t}$ and $W_{g,t}$ denote the local and global model parameters, respectively. The operator $\mathrm{flat}(\cdot)$ concatenates all matched model parameters into a one-dimensional vector. We treat the high-dimensional update vector $\Delta W_{i,t}$ as samples from an empirical random variable X_i. The vector is approximated by a histogram with B bins, producing a discrete probability distribution P_i. Similarly, the global reference update is converted into a reference distribution P_g using shared bin edges. This allows the server to compare local and global update distributions under a common empirical support.

B.2. Information-Theoretic Metrics

To characterize each client update from multiple perspectives, we extract three information-theoretic indicators:

$$\mathbf{e}_{i,t} = [H(P_i), D_{KL}(P_i \parallel P_g), I(X_i; X_g)].$$

Entropy. Entropy measures the uncertainty or dispersion of the client update distribution:

$$H(P_i) = -\sum_{x \in \mathcal{X}} P_i(x) \log P_i(x).$$

In our setting, P_i is the histogram-based empirical distribution of $\Delta W_{i,t}$. Poisoning or noisy updates may alter the dispersion pattern of the update distribution; therefore, entropy is used as a complementary signal for detecting abnormal update behavior.

KL-divergence. KL-divergence measures the relative divergence of the local update distribution P_i from the global reference distribution P_g:

$$D_{KL}(P_i \parallel P_g) = \sum_{x \in \mathcal{X}} P_i(x) \log \frac{P_i(x)}{P_g(x)}.$$

This metric captures how much a client update deviates from the global reference update distribution and serves as a signal for detecting distribution-shifting abnormal updates, such as poisoning-induced deviations.

Mutual information. Mutual information measures the statistical dependency between the local update and the global reference update:

$$I(X_i; X_g) = \sum_{x,y} P_{i,g}(x, y) \log \frac{P_{i,g}(x, y)}{P_i(x) P_g(y)}.$$

Here, $P_{i,g}$ denotes the joint probability mass function estimated from a two-dimensional histogram over aligned update indices. This metric indicates whether the client update follows a pattern statistically aligned with the global reference update, rather than measuring mutual information with task labels.

B.3. Robust Evidence Normalization

The information-theoretic indicators are first combined into a scalar evidence score:

$$S_{\text{raw},i} = \omega_1 I(X_i; X_g) - \omega_2 D_{KL}(P_i \parallel P_g) - \omega_3 \left| H(P_i) - \bar{H} \right|,$$

where $\bar{H}$ is the mean entropy of participating clients in the current round. The score rewards updates aligned with the global reference and penalizes large divergence or abnormal entropy deviation.

To reduce sensitivity to outliers, we normalize the raw evidence scores using median absolute deviation (MAD):

$$z_{i,t} = \frac{S_{\text{raw},i} - \text{median}(\mathbf{S}_{\text{raw}})}{1.4826 \cdot \max(\text{MAD}(\mathbf{S}_{\text{raw}}), 1.0)}.$$

Here, $\mathbf{S}_{\text{raw}}$ denotes the set of raw scores from participating clients. The coefficient 1.4826 makes MAD comparable to the standard deviation under a normal distribution, and the lower bound prevents excessive amplification when client updates are similar. The normalized score $z_{i,t}$ is used only as behavioral evidence for trust updating, not as the final trust score.

B.4. Bayesian Trust Accumulation and Dynamic State Management

For each client, the server maintains a Bayesian trust ledger based on a Beta distribution:

$$T_{i,t} = \mathbb{E}[\text{Beta}(\alpha_{i,t}, \beta_{i,t})] = \frac{\alpha_{i,t}}{\alpha_{i,t} + \beta_{i,t}}.$$

The parameter $\alpha_{i,t}$ represents accumulated success evidence, while $\beta_{i,t}$ represents accumulated failure evidence. All clients are initialized with the same prior $\mathrm{Beta}(\alpha_0, \beta_0)$.

At each round, the robust evidence score $z_{i,t}$ is converted into a binary behavioral outcome using a dual-gate rule:

$$y_{i,t} = \begin{cases} 0, & z_{i,t} < q_P(\mathbf{z}_t) \text{ and } z_{i,t} < z_{\mathrm{abs}}, \\ 1, & \text{otherwise}, \end{cases}$$

where $\mathbf{z}_t$ is the set of evidence scores of participating clients, $q_P(\mathbf{z}_t)$ is the P-th percentile cutoff, and z_{abs} is an absolute anomaly threshold. The trust parameters are then updated as

$$\alpha_{i,t+1} = \alpha_{i,t} + y_{i,t}, \qquad \beta_{i,t+1} = \beta_{i,t} + (1 - y_{i,t})\kappa,$$

where κ controls the penalty strength for failure evidence. Thus, a client is penalized only when its evidence is both relatively low among participants and absolutely anomalous.

For non-participating clients, the trust state gradually regresses toward the neutral prior:

$$\alpha_{i,t+1} = (\alpha_{i,t} - 1)\delta + 1, \qquad \beta_{i,t+1} = (\beta_{i,t} - 1)\delta + 1,$$

where δ is the non-participation decay factor. This prevents outdated trust values from being preserved indefinitely in dynamic IoT participation.

Finally, the server maintains a strike counter for persistent suspicious behavior. A strike is assigned only when both accumulated trust and current evidence indicate abnormality:

$$T_{i,t} < \tau_t \quad \text{and} \quad z_{i,t} < z_{\mathrm{abs}}.$$

If this condition holds for M consecutive participations, the client is hard-dropped from future aggregation. This rule separates persistent malicious behavior from temporary instability or benign non-IID deviation.

Appendix C. LLM-Agent-Guided Stackelberg Policy Orchestration

C.1. Stackelberg Game Formulation

We formulate the interaction between the FL server and participating clients as a Stackelberg game. The server acts as the leader, while clients act as followers. At each round t, the server determines policy variables that regulate trust thresholding, reward allocation, and aggregation behavior. Each client then decides whether to participate based on its expected reward and local participation cost.

The server utility is defined as

$$U^t_{\mathrm{server}} = \alpha_t \Delta\mathrm{Acc}_t - \beta_t \sum_{i \in S_t} r_t(i),$$

where $\Delta\mathrm{Acc}_t$ denotes the improvement in global model accuracy, $r_t(i)$ is the reward assigned to client i, and α_t and β_t control the trade-off between model improvement and reward cost. The client utility is defined as

$$U^t_{\mathrm{client},i} = r_t(i) - c^t_i,$$

where c^t_i denotes the local cost of participation. A client is incentivized to participate when its expected reward exceeds its local cost.

C.2. LLM Agent as a Bounded Threat Assessor

In the proposed framework, the LLM agent serves as a high-level threat assessor rather than a direct controller. To avoid unstable or unreproducible LLM-driven control, the agent does not directly output aggregation weights, trust scores, client exclusion decisions, or policy parameters. Instead, it reads recent system-level statistics and outputs a single bounded scalar threat level:

$$\rho_t \in [0, 1].$$

The input to the LLM agent consists of a fixed-size history window of recent training summaries. Each summary includes round-level statistics such as macro-F1, global accuracy, average trust, trust standard deviation, participation churn, number of participants, number of dropped clients, the previously used threshold, and the previously used threat level. A low threat level indicates stable performance and uniform trust, whereas a high threat level indicates possible attack behavior, performance degradation, increasing trust dispersion, or excessive client drops.

The agent is required to return a strict JSON object containing a short reasoning field and a scalar threat level. The reasoning field is stored for interpretability, but only ρ_t is used for policy computation. Thus, the LLM provides strategic state assessment, while all numerical policy variables are computed outside the prompt through deterministic formulas.

C.3. Formula-Driven Policy Mapping

After obtaining the threat level ρ_t, the framework maps it to Stackelberg policy parameters using fixed monotone transformations:

$$\gamma_t = \gamma_{\min} + (\gamma_{\max} - \gamma_{\min})\rho_t,$$

$$\lambda_t = \lambda_{\min} + (\lambda_{\max} - \lambda_{\min})\rho_t,$$

$$\alpha_t = \alpha_{\min} + (\alpha_{\max} - \alpha_{\min})\rho_t, \qquad \beta_t = 1 - \alpha_t.$$

Here, γ_t controls trust-threshold sensitivity, λ_t controls reward decay for low-trust clients, and α_t, β_t determine the accuracy-cost trade-off in the server utility. Since all parameters are derived from fixed formulas, the final defense behavior remains transparent and reproducible.

The dynamic trust threshold is computed from the current trust distribution:

$$\tau_t = \text{clip}\left(\mu_T^t + \gamma_t \sigma_T^t, \tau_{\min}, \tau_{\max}\right),$$

where μ_T^t and σ_T^t denote the mean and standard deviation of client trust scores at round t. As ρ_t increases, γ_t increases, making the threshold more conservative under high-risk conditions.

C.4. Reward and Participation Rule

The reward assigned to client i is computed as

$$r_t(i) = R_{\text{base}} \exp\left(-\lambda_t(1 - T_i^t)\right).$$

This reward function gives higher rewards to high-trust clients and exponentially suppresses rewards for low-trust clients. When the threat level is high, λ_t increases, causing stronger reward decay for suspicious clients.

Given the reward $r_t(i)$, client i participates only when its utility is positive:

$$a_i^t = \begin{cases} 1, & r_t(i) - c_i^t > 0, \\ 0, & \text{otherwise.} \end{cases}$$

Thus, low-trust clients can be excluded not only by aggregation filtering but also by incentive-aware participation control.

C.5. Trust-Aware Aggregation and Failure Handling

The final aggregation weight is determined by the Bayesian trust score and the current policy threshold. A client contributes to aggregation only if it is not hard-dropped and its trust score satisfies the current threshold:

$$w_i^t = \begin{cases} \dfrac{T_i^t \sqrt{n_i}}{\sum_{j \in \mathcal{A}_t} T_j^t \sqrt{n_j}}, & i \in \mathcal{A}_t, \\ 0, & \text{otherwise,} \end{cases}$$

where

$$\mathcal{A}_t = \{ j \in S_t \mid T_j^t \geq \tau_t,\ j \notin \mathcal{D}_t \}$$

denotes the accepted client set. The global model is then updated by trust-weighted aggregation over accepted clients.

To prevent invalid LLM outputs from destabilizing training, the framework includes a fallback mechanism. During cold-start rounds or when no history is available, the system uses the last valid policy. If the LLM call fails, returns an unparsable response, or produces an invalid threat level, the framework reuses the previous valid policy. Therefore, the LLM agent provides adaptive threat assessment, while the actual Stackelberg policy remains numerically constrained by fixed mappings and Bayesian trust scores.

Appendix D. Full Algorithm of Trust-aware Stackelberg Federated Learning

Algorithm 1: Trust-aware Stackelberg Federated Learning (TS-FL)

Data: w_t; client set $\mathcal{C}$; participants S_t; trust ledger (α_i, β_i); strikes m_i; dropped set $\mathcal{D}_{t-1}$; history $\mathcal{H}_{t-1}$
Result: Updated model w_{t+1}, trust ledger, and dropped set $\mathcal{D}_t$
$\mathcal{D}_t \leftarrow \mathcal{D}_{t-1}$

```
// 1. Local update and information-theoretic evidence
```

for $i \in S_t$ *in parallel* **do**
  Train locally to obtain $w_{i,t}$, $\Delta w_{i,t} \leftarrow w_{i,t} - w_t$, estimate P_i from $\Delta w_{i,t}$, $\mathbf{e}_{i,t} \leftarrow [H_i, D_{KL,i}, I_i]$
end

```
// 2. Robust evidence scoring and Bayesian trust update
```

Compute $S_{\text{raw},i}$ for all $i \in S_t$, and normalize them into $\{z_{i,t}\}_{i \in S_t}$ using MAD
for $i \in S_t$ **do**
  if $z_{i,t} < q_P(\mathbf{z}_t)$ *and* $z_{i,t} < z_{\text{abs}}$ **then**
    $\beta_i \leftarrow \beta_i + \kappa$, $m_i \leftarrow m_i + 1$
  else
    $\alpha_i \leftarrow \alpha_i + 1$, $m_i \leftarrow 0$
  end
  $T_{i,t} \leftarrow \alpha_i/(\alpha_i + \beta_i)$; **if** $m_i \geq M$ **then**
    $\mathcal{D}_t \leftarrow \mathcal{D}_t \cup \{i\}$
  end
end

```
// 3. Trust regression for non-participating clients
```

for $i \in \mathcal{C} \setminus S_t$ **do**
  $\alpha_i \leftarrow (\alpha_i - 1)\delta + 1$, $\beta_i \leftarrow (\beta_i - 1)\delta + 1$, $T_{i,t} \leftarrow \alpha_i/(\alpha_i + \beta_i)$
end

```
// 4. LLM-guided Stackelberg policy mapping
```

$\rho_t \leftarrow \text{LLM_Agent}(\mathcal{H}_{t-1})$
if ρ_t *is invalid* **then**
  $\rho_t \leftarrow \rho_{t-1}$
end
$(\gamma_t, \lambda_t, \alpha_t, \beta_t) \leftarrow f_{\text{policy}}(\rho_t)$, $\tau_t \leftarrow \text{clip}(\mu_T^t + \gamma_t \sigma_T^t, \tau_{\min}, \tau_{\max})$

```
// 5. Reward-aware filtering
```

for $i \in S_t$ **do**
  $r_t(i) \leftarrow R_{\text{base}} \exp[-\lambda_t(1 - T_{i,t})]$, $U_i^t \leftarrow r_t(i) - c_i^t$
end
$\mathcal{A}_t \leftarrow \{i \in S_t \mid T_{i,t} \geq \tau_t,\ i \notin \mathcal{D}_t,\ U_i^t > 0\}$

```
// 6. Trust-weighted aggregation
```

for $i \in \mathcal{A}_t$ **do**
  $\omega_{i,t}^{\text{agg}} \leftarrow \dfrac{T_{i,t}\sqrt{n_i}}{\sum_{j \in \mathcal{A}_t} T_{j,t}\sqrt{n_j}}$
end
$w_{t+1} \leftarrow w_t + \sum_{i \in \mathcal{A}_t} \omega_{i,t}^{\text{agg}}(w_{i,t} - w_t)$, $\mathcal{H}_t \leftarrow \mathcal{H}_{t-1} \cup \text{summary}_t$
return w_{t+1}, $\{(\alpha_i, \beta_i)\}_{i=1}^N$, $\mathcal{D}_t$

Appendix E. Experimental Details

Table 4: Summary of experimental configurations.

Configuration	Value
Dataset	X-IIoTID
Task	Multi-class NIDS using `class1`
Train/test split	80:20
Number of clients	20
Communication rounds	200
Local model	MLP $(256, 128, 64)$
Local epochs	2
Optimizer	Adam
Learning rate	10^{-3}
Batch size	128
Client availability	0.7
Attack phases	P1: 1–40, P2: 41–120, P3: 121–160, P4: 161–200
Baselines	FedAvg, Krum, Median, Trimmed Mean
LLM model	`qwen3:14b` via Ollama

E.1. Dataset and Preprocessing

We evaluate the proposed framework on the X-IIoTID dataset, which contains heterogeneous IoT network traffic records for intrusion detection. The task is formulated as a multi-class network intrusion detection problem using the `class1` label, where each class corresponds to either normal traffic or a specific attack category. Samples with missing values were removed, and categorical attributes were converted into numerical values using label encoding. To focus on trustworthy federated learning under dynamic adversarial participation rather than extreme data scarcity, attack classes with fewer than 2,000 training samples were excluded. The dataset was split into training and test sets with an 80:20 ratio.

E.2. Federated Learning Setting

We simulate a cross-device FL environment with 20 clients over 200 communication rounds. Each client trains a local multilayer perceptron model with hidden dimensions $(256, 128, 64)$ for two local epochs using the Adam optimizer with a learning rate of 10^{-3} and a batch size of 128.

To reflect practical IoT data heterogeneity, we employ a hybrid non-IID partitioning strategy. Each client receives 3,000 normal samples and is additionally assigned a random number of attack classes, sampled uniformly between one and six, with 1,000 samples per assigned attack class. Each attack class appears in at least three clients to avoid unrealistically isolated class ownership. Client participation is stochastic, with an availability probability of 0.7 in each round.

E.3. Dynamic Attack Scenario

To evaluate robustness under dynamic IoT participation, we introduce time-varying attack waves. The first 40 rounds are clean rounds, allowing benign convergence and trust accumulation. From

rounds 41 to 120, malicious clients become active and perform coordinated model-shift attacks. Rounds 121 to 160 represent a dormant phase in which attackers behave benignly, followed by a second attack wave from rounds 161 to 200.

This setting tests whether a defense can suppress the first attack wave and prevent previously detected malicious clients from re-entering the aggregation process during later attack resurgence.

E.4. Policy Hyperparameters

Table 5 summarizes the main hyperparameters used in the proposed framework. The LLM agent outputs only a bounded threat level $\rho_t \in [0, 1]$, which is converted into policy parameters through deterministic bounded mappings. For the information-theoretic evidence score in Eq. (1), we set $\omega_1 = \omega_2 = \omega_3 = 1.0$, assigning equal importance to mutual information, KL divergence, and entropy deviation.

Table 5: Policy and trust hyperparameters used in the proposed framework.

Parameter	Value	Description
LLM-guided policy mapping		
ρ_t	$[0, 1]$	LLM threat level
ρ_0	0.20	Default threat level before LLM activation
γ_t	$0.20 + 0.60\rho_t$	Trust-threshold sensitivity
λ_t	$0.10 + 1.90\rho_t$	Reward-decay coefficient
α_t	$0.55 + 0.35\rho_t$	Accuracy weight in server utility
β_t	$1 - \alpha_t$	Reward-cost weight in server utility
τ_t	$\mathrm{clip}(\mu_T^t + \gamma_t \sigma_T^t, 0.35, 0.85)$	Trust threshold
$r_t(i)$	$R_{\text{base}} \exp[-\lambda_t(1 - T_{i,t})]$	Client reward
R_{base}	100.0	Base reward for participation
Information-theoretic evidence and trust update		
$\omega_1, \omega_2, \omega_3$	$1.0, 1.0, 1.0$	Weights for I_i, $D_{KL,i}$, and $\lvert H_i - \bar{H} \rvert$
B	50	Number of histogram bins
P	50	Percentile threshold for anomaly gate
$q_P(\mathbf{z}_t)$	$q_{50}(\mathbf{z}_t)$	Round-wise percentile threshold over normalized scores
z_{abs}	-1.5	Absolute MAD-normalized anomaly threshold
κ	3.0	Failure penalty added to the Beta trust posterior
α_0, β_0	$1.0, 1.0$	Initial Beta prior parameters
δ	0.999	Non-participant trust decay factor
M	5	Maximum consecutive strikes before hard drop
LLM invocation		
t_{LLM}	4	First round in which the LLM agent is queried
W_{LLM}	10	Recent-round observation window size

E.5. Implementation Details

All experiments are implemented in Python using PyTorch 2.5.1 with CUDA 12.1. The random seed is fixed to 42 for reproducibility. Experiments are conducted on a workstation with an NVIDIA GeForce RTX 3090 GPU, a 12th Gen Intel Core i7-12700K CPU, and 32GB RAM. The LLM agent

is implemented through a locally hosted Ollama `qwen3:14b` model, which is queried after the initial warm-up rounds using recent trust and performance observations.

Appendix F. Additional Experiments

This appendix reports three additional experiments to examine the robustness and sensitivity of the proposed framework. First, we evaluate the framework across multiple attack scenarios, including coordinated shift, free-rider, model poisoning, and sign-flip attacks. Second, we compare different LLM backbones for threat-level estimation. Third, we vary the number of participating clients. In all cases, the overall performance trends remain stable, suggesting that the proposed framework is not strongly dependent on a specific attack type, LLM backbone, or client population size.

F.1. Robustness across Attack Scenarios

To verify that the proposed defense is not over-tuned to a single threat, we evaluate it under coordinated-shift, free-rider, model-poisoning, and sign-flip attacks. As summarized in Table 6, the proposed method maintains stable performance across all scenarios. Coordinated-shift is the most disruptive setting, producing the lowest P2 macro-F1 score of 0.7485; however, the performance recovers to 0.8157 by P4. In contrast, under free-rider, model-poisoning, and sign-flip attacks, the framework maintains relatively high P2 macro-F1 scores of 0.8092, 0.7966, and 0.7988, respectively. These results suggest that the proposed trust-based adaptive filtering generalizes effectively across diverse adversarial behaviors, rather than being specialized to a single attack type.

Table 6: Phase-level performance of Ours under different attack scenarios.

Attack	**P1 Clean**		**P2 Attack**		**P3 Clean**		**P4 Attack**	
	Acc	F1	Acc	F1	Acc	F1	Acc	F1
Coordinated Shift	0.9550	0.7609	0.9414	0.7485	0.9828	0.8058	0.9855	0.8157
Free-rider	0.9508	0.7530	0.9816	0.8092	0.9811	0.7934	0.9738	0.7698
Model Poison	0.9510	0.7533	0.9741	0.7966	0.9835	0.8085	0.9857	0.8154
Sign Flip	0.9517	0.7546	0.9734	0.7988	0.9839	0.8116	0.9861	0.8209

F.2. Effect of Different LLM Backbones

To examine whether the proposed framework depends heavily on a particular LLM, we compare several backbone models for threat-level estimation, including `llama3.1:8b`, `qwen3:14b`, `gpt4o`, `gemma3:12b`, and `mistral:7b`. Figure 4 shows the round-wise accuracy and macro-F1 trajectories.

Overall, all LLM variants exhibit highly similar behavior. The models show nearly overlapping convergence trends in the clean phase, a sharp short-term degradation at the beginning of the first attack wave, and subsequent recovery and stabilization. While small fluctuations are observed in some rounds, especially in macro-F1, no substantial performance gap is observed across the tested LLM backbones. These results suggest that the proposed framework is not strongly tied to a specific LLM model, as long as the agent can provide a reasonable bounded threat estimate.

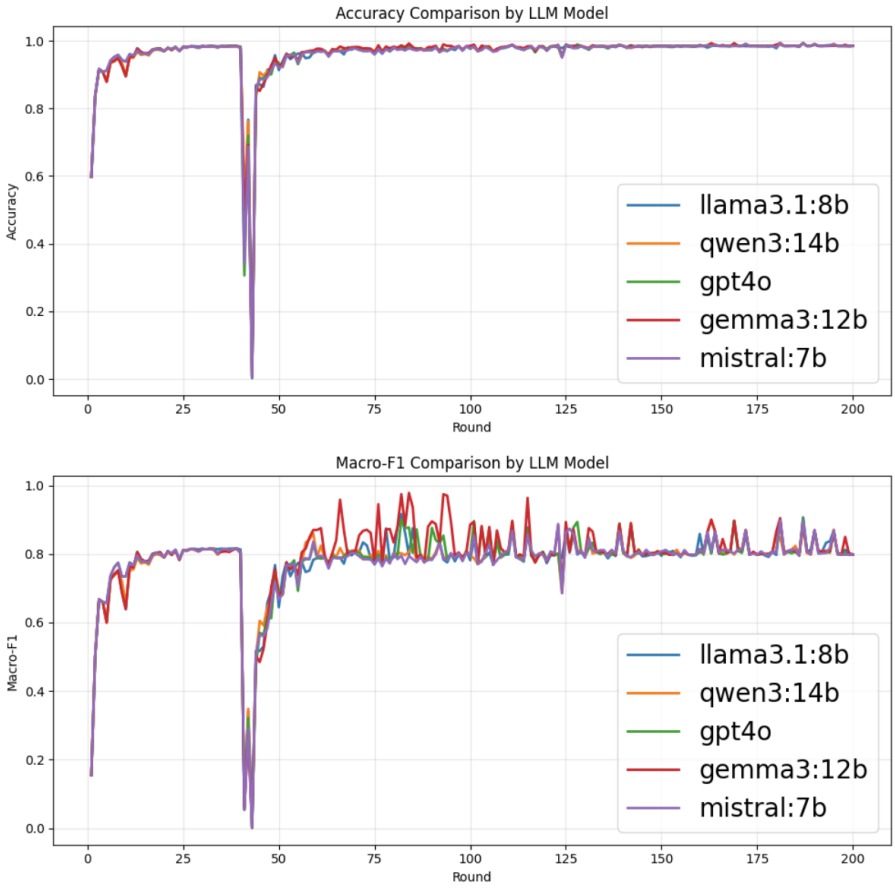

Figure 4: Comparison of the proposed framework using different LLM backbones. Top: round-wise accuracy. Bottom: round-wise macro-F1. The overall trajectories are highly similar across models.

F.3. Effect of the Number of Clients

We also evaluate the effect of changing the number of clients. Specifically, we compare the proposed method under different client populations, $N \in \{10, 20, 30, 40, 50\}$. Figure 5 shows the round-wise macro-F1 trajectories under the coordinated-shift attack setting.

The overall trends remain similar across different client counts. All settings show stable convergence in the clean phase, a short-term drop near the onset of the attack phase, and subsequent recovery to a similar performance range. Although minor variations appear in the magnitude of temporary fluctuations, no meaningful performance degradation is observed as the number of clients changes. This indicates that the proposed framework maintains consistent behavior under moderate changes in the scale of participating clients.

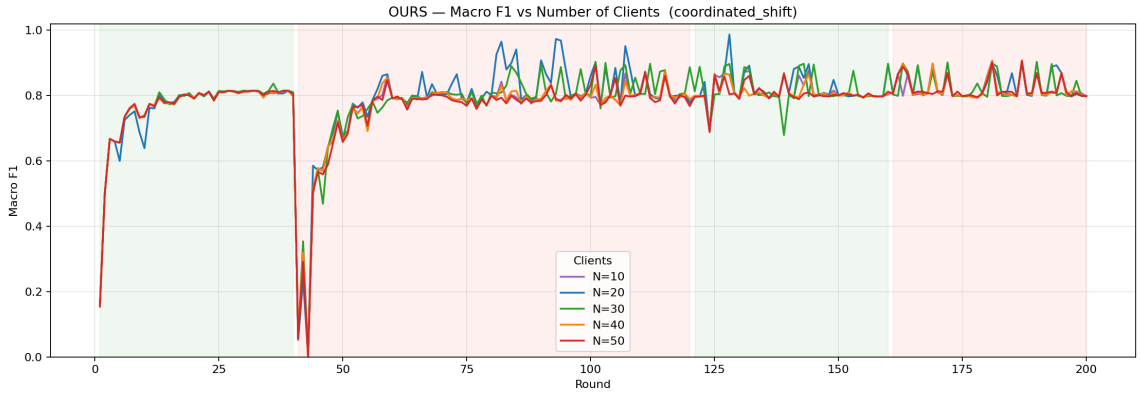

Figure 5: Macro-F1 comparison under different numbers of clients ($N = 10, 20, 30, 40, 50$) in the coordinated-shift attack setting. The overall trends remain similar across client counts.

Appendix G. Detailed Analysis of LLM-Guided Policy and Trust Stability

G.1. LLM-Guided Policy Adaptation

This appendix reports additional round-wise evidence for the adaptive behavior of the proposed framework. The LLM agent acts only as a bounded threat assessor; it does not directly generate aggregation weights, trust scores, or exclusion decisions.

Before the first attack wave, the system remains stable with no dropped clients, and the threat level remains low. Near the end of P1, the model reaches a macro-F1 of approximately 0.8057 with an average trust score of 0.964. When the first coordinated-shift attack begins, the macro-F1 drops to 0.0639, indicating a non-trivial short-term degradation.

After repeated degradation is observed, the LLM agent increases the threat level from the low-threat regime to $\rho_t = 0.80$ at round 42 and $\rho_t = 1.00$ at round 44. Through the deterministic policy mapping, these values increase trust-threshold sensitivity and reward decay. The natural-language rationale is stored for interpretability, but only the scalar threat level ρ_t is used for policy computation.

Table 7: Representative LLM-agent responses during the coordinated-shift attack.

Round	$\boldsymbol{\rho}$	Observed Evidence	Policy Interpretation
40	0.20	Stable F1, high trust, no drops	Low-threat regime
41	0.20	First abrupt F1 collapse	Avoid one-shot overreaction
42	0.80	Severe F1 degradation	High-threat response
44	1.00	Near-zero F1 and trust dispersion	Crisis-level response
57–71	0.20–0.60	F1 recovery with increasing drops	Stabilized adaptive filtering

G.2. Trust Separation and Malicious-Client Removal

Table 8: Average trust scores of benign and malicious clients in Ours.

Phase	Benign Trust	Malicious Trust	Avg. Gap	Max Gap
P1 Clean	0.8773	0.8806	-0.0034	0.0382
P2 Attack	0.9800	0.6401	0.3399	0.3765
P3 Clean	0.9893	0.6112	0.3781	0.3795
P4 Attack	0.9917	0.6112	0.3804	0.3813

Table 8 shows that the proposed trust mechanism separates benign and malicious clients after the first attack wave begins. During P1, benign and malicious clients have almost identical trust scores because malicious clients behave benignly during the clean phase. The average trust gap is only -0.0034 in P1. After the attack begins, benign trust increases to 0.9800 in P2, while malicious trust decreases to 0.6401. This gap persists in P3 and P4, indicating that the system retains memory of previously suspicious behavior even when the attack becomes dormant.

Table 9: Drop analysis of the proposed method.

Phase	New M	New B	Cum. Drop	Cum. M	Cum. B	Prec.	Recall
P1 Clean	0	0	0	0	0	N/A	0.000
P2 Attack	8	0	8	8	0	1.000	1.000
P3 Clean	0	0	8	8	0	1.000	1.000
P4 Attack	0	0	8	8	0	1.000	1.000

Table 9 shows that all malicious clients are removed during the first attack wave P2. No additional clients are dropped in P3 or P4, indicating that the defense has already isolated the malicious pool before the second attack wave. Therefore, the robustness observed in P4 is due to persistent trust memory and hard-drop exclusion, rather than repeated re-detection during the second attack wave.

We distinguish between scheduled attack activation and effective malicious participation. In P4, the attack schedule activates the same malicious pool, but effective malicious participation is blocked because those clients were already hard-dropped in P2. This setting evaluates whether the defense retains memory and prevents recurring attackers from regaining influence.

Overall, these results support the intended adaptive-defense behavior. The initial attack succeeds and causes a sharp performance collapse, confirming that the threat scenario is non-trivial. The LLM agent then increases the threat level based on round-wise evidence, and the Bayesian trust mechanism progressively removes suspicious clients. After all malicious clients are removed during P2, the system recovers during P3. When the malicious pool is scheduled to reappear in P4, the previously identified attackers remain hard-dropped and cannot participate in aggregation. Thus, the stability in P4 reflects persistent exclusion through trust memory, rather than repeated detection from scratch.

G.3. Computational Efficiency Analysis

To evaluate the operational feasibility of the proposed framework, we measured the average execution time per communication round on a server equipped with an Ollama-hosted $qwen3:14b$ model. Table 10 compares the per-round latency of TS-FL against conventional robust aggregation baselines.

Table 10: Average execution time per round (seconds).

Method	Ours	FedAvg	Krum	Trimmed Mean	Median
Time (s)	5.903	3.814	3.836	3.823	3.961

The proposed framework requires 5.903 seconds per round on average, whereas FedAvg requires 3.814 seconds, resulting in a measured round-level increase of approximately 2.089 seconds. The LLM inference latency was measured over 197 valid invocation rounds because the first three warm-up rounds do not trigger LLM calls. Its average latency is 3.191 seconds, corresponding to approximately 54% of the total TS-FL round time.

The measured round-level increase relative to FedAvg should not be interpreted as the LLM latency alone. While the LLM call is the largest additional server-side component, the end-to-end round time is also affected by trust-based client filtering, the number of accepted clients, aggregation workload, and execution variance. Thus, the 2.089-second increase represents the observed end-to-end latency gap, whereas the 3.191-second value represents the average cost of each LLM invocation.

This overhead is acceptable for security-oriented IoT FL because the LLM is used only for high-level state assessment and does not process client model weights directly. In addition, the full TS-FL model improves macro-F1 by 7.1 percentage points over the variant without LLM orchestration, indicating that the additional computation contributes to adaptive robustness.