Federated Semi-Supervised and Semi-Asynchronous Learning for Anomaly Detection in IoT Networks

Liang Liu, Wenbin Zhai, Feng Wang, Youwei Ding, Wanying Lu, and Weizhi Meng, *Senior Member, IEEE*

***Abstract*—The expansive attack surfaces and device heterogeneity of Internet of Things (IoT) networks pose significant challenges for anomaly detection. While Federated Learning (FL) enables privacy-preserving detection, existing FL methods typically assume fully labeled client data, which is unrealistic for practical IoT deployments. Resource constraints and network heterogeneity further complicate the trade-off among training efficiency, detection accuracy, and communication overhead. To address these challenges, we propose FedS^{3}A, a Federated Semi-Supervised and Semi-Asynchronous learning framework for IoT anomaly detection. FedS^{3}A operates in a practical disjoint semi-supervised setting where the server holds limited labeled data and clients possess extensive unlabeled data. We apply pseudo-labeling with a dynamically decaying weight to balance server-side supervised training and client-side unsupervised learning. To improve round efficiency, we introduce a semi-asynchronous model update and staleness-tolerant distribution scheme that scales client contributions to the global model based on local model staleness and participation frequency. We also adopt a group-based aggregation function to mitigate the impact of non-IID client data, and utilize sparse difference transmission to reduce communication overhead. We evaluate FedS^{3}A on the CIC-IDS2017 and Edge-IIoTset datasets, and the results demonstrate that FedS^{3}A consistently outperforms representative FL approaches in detection performance and round efficiency. FedS^{3}A achieves over 98% accuracy even under non-IID settings while reducing communication costs by approximately 50%.**

***Index Terms*—Internet of Things, anomaly detection, federated learning, semi-supervised learning, semi-asynchronous learning.**

I. Introduction

OVER the past decades, advancements in sensor and wireless communication technologies have established Internet of Things (IoT) networks as the foundational architecture for numerous applications, including smart homes, healthcare, cities, grids, and transportation [1]. However, the massive deployment of IoT networks has significantly expanded their attack surface, rendering devices vulnerable to Distributed Denial-of-Service (DDoS), side-channel, and botnet attacks [2], [3], [4]. These threats significantly hinder the widespread adoption of IoT technologies [5], making effective countermeasures for detecting compromised devices critical.

Machine Learning (ML) and Deep Learning (DL) techniques are now widely adopted for IoT attack detection due to their superior performance [6], [7], [8]. Traditional methods are typically categorized into centralized and distributed detection architectures [9]. In centralized detection, IoT devices transmit local data to a central server. While this approach achieves high accuracy, it introduces substantial communication costs, privacy risks, and limited scalability. Distributed detection performs inference locally using private data. However, the lack of collaboration across devices creates isolated data silos, often resulting in lower detection accuracy [10].

To address these limitations, Federated Learning (FL) [16] enables multiple clients to collaboratively train a global model under a central server without sharing raw local data. During each training round, clients train locally and upload only model parameters. The server then aggregates and redistributes these parameters to consolidate knowledge across clients. This process balances privacy, accuracy, communication overhead, and latency.

The FL procedure continues until model convergence or a specified number of rounds is reached. While numerous FL-based approaches exist for IoT anomaly detection, as summarized in Table I [11], [12], [13], [14], [15], they still face several critical challenges:

- **Lack of Labeled Data:** Existing works typically rely on supervised learning, where local data is fully annotated. This assumption is unrealistic because end users rarely possess the expertise to accurately label network traffic [17]. Although some studies [11], [15] claim to use unsupervised schemes, they assume all training data is benign. This implicit assumption essentially reduces the problem to a supervised setting. Furthermore, if devices are compromised during the training phase, malicious samples will poison the model and degrade detection performance [18].
- **Device Heterogeneity:** In synchronous FL [11], [19], [20], differences in computing capacity, network bandwidth, and data volume cause significant variations in local update times [21]. Consequently, the server must wait for the slowest client, leading to idle time and

Received 12 May 2025; revised 18 April 2026 and 26 July 2026; accepted 12 August 2026. Date of publication 25 August 2026; date of current version 8 September 2026. This work was supported by the National Science and Technology Major Project of China (No. 2025ZD1600800). The associate editor coordinating the review of this article and approving it for publication was A. Lahmadi. *(Liang Liu and Wenbin Zhai contributed equally to this work.) (Corresponding author: Weizhi Meng.)*

Liang Liu is with the College of Software, Nankai University, Tianjin 300071, China (e-mail: ll@nankai.edu.cn).

Wenbin Zhai is with the Department of Computing, The Hong Kong Polytechnic University, Hong Kong (e-mail: wenbin.zhai@connect.polyu.hk).

Feng Wang and Wanying Lu are with the College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing 210016, China (e-mail: fengwang06@nuaa.edu.cn; wanyinglu@nuaa.edu.cn).

Youwei Ding is with the School of Artificial Intelligence and Information Technology, Nanjing University of Chinese Medicine, Nanjing 210023, China (e-mail: ywding@njucm.edu.cn).

Weizhi Meng is with the School of Computing and Communications, Lancaster University, LA1 4WA Lancaster, U.K. (e-mail: weizhi.meng@ieee.org).

Digital Object Identifier 10.1109/TNSM.2026.3727228

low round efficiency. Asynchronous FL mitigates this problem but introduces frequent model updates. Processing continuous local updates can overwhelm the server and consume excessive communication resources while providing marginal benefits to model convergence [22].

- **Non-IID Data:** In IoT environments, devices exhibit diverse behaviors, and only a subset may experience attacks. Thus, local data across devices rarely represents a uniform sample distribution. This results in highly non-Independent and Identically Distributed (non-IID) data across clients [23]. Prior research shows that non-IID data severely degrades both convergence rates and model accuracy [24].
- **Significant Communication Cost:** While FL eliminates raw data transmission, exchanging model parameters over numerous communication rounds still imposes substantial overhead on resource-constrained IoT networks [25]. Additionally, the central server can become a communication bottleneck when processing concurrent model transfers from many clients [26].

To address these challenges, we propose FedS^{3}A, a Federated Semi-Supervised and Semi-Asynchronous learning framework for anomaly detection in IoT networks. FedS^{3}A operates under a realistic disjoint data setting where the central server holds limited labeled data and numerous clients possess extensive unlabeled data. During each training round, clients perform unsupervised local training guided by a pseudo-labeling technique. To accommodate device heterogeneity, the server triggers a global model update semi-asynchronously once it receives parameters from a predefined fraction of clients. The server then applies a specialized aggregation function to mitigate the adverse effects of stale models and non-IID data distributions. Finally, FedS^{3}A employs sparse difference transmission to minimize bandwidth consumption during parameter exchanges. The main contributions of this paper are summarized as follows:

- We formulate a practical federated semi-supervised anomaly detection system for IoT networks. We design a pseudo-labeling mechanism with dynamic weights to integrate server-side supervised learning seamlessly with client-side unsupervised training.
- We introduce a semi-asynchronous update mechanism paired with a specialized aggregation function. This design handles device heterogeneity by tolerating model staleness and incorporates a group-based aggregation strategy to capture dominant gradient features under non-IID data distributions. We develop a sparse difference transmission method to optimize communication efficiency.
- Extensive evaluations on the CIC-IDS2017 [27] and Edge-IIoTset [28] datasets demonstrate that FedS^{3}A achieves over 98% accuracy under non-IID settings and reduces communication costs by approximately 50% compared to representative FL baselines [11], [12], [13], [14], [15].

The remainder of this paper is organized as follows. Section II reviews related work. Section III introduces the FL procedure and formalizes the system model. Section IV details the design of FedS^{3}A. Section V evaluates the performance of FedS^{3}A. Finally, Section VI concludes the paper.

II. Related Work

A. Federated Anomaly Detection in IoT Networks

Federated learning is widely adopted for IoT intrusion and anomaly detection to enable collaborative training without exposing raw traffic data. Early studies established the feasibility of this approach. DIoT [11] detects IoT device compromise by modeling benign behavior patterns via recurrent architectures. Liu et al. [14] combine CNN-based feature extraction with LSTM-based detection and use top-k gradient compression to reduce communication overhead. Rey et al. [12] study supervised and benign-only settings on N-BaIoT. Campos et al. [13] investigate client distribution scenarios on CIC-ToN-IoT and propose the Fed+ aggregation rule for skewed data. Tian et al. [15] address gradient divergence and delay induced by device heterogeneity and non-IID data.

Recent works extend this to address emerging challenges. FL-IIDS [29] introduces incremental federated intrusion detection to mitigate catastrophic forgetting of new attack classes. FedKD-IDS [30] replaces direct parameter aggregation with knowledge distillation and anti-poisoning verification. FTKD [31] explores two-stage knowledge distillation for industrial IoT. While these studies strengthen federated IDS design, they focus on standard collaborative protocols. They do not address our joint setting where client-side data are unlabeled, supervision is concentrated at the server, and training must tolerate stale and partial client participation.

B. Label-Efficient Federated Learning

A second line of work addresses labeled data scarcity in federated environments. Semi-supervised learning [32] utilizes abundant unlabeled data alongside limited supervision. Its federated extension has become a critical research direction [33]. Existing studies typically distinguish between labels-at-client and labels-at-server settings based on supervision availability [17]. Early federated semi-supervised learning research focuses on pseudo-labeling, data augmentation, and aggregation strategies. Zhu et al. [34] combine CNN-GRU with pseudo-labeling, group-based aggregation, and flip-based augmentation. Nandury et al. [35] analyze update diversity and introduce diversity scaling. Zhang et al. [36] study non-IID data effects and design group-based aggregation to balance supervised and unsupervised objectives.

Recent works address challenging labels-at-server regimes with severe label deficiency. Twin-Sight [37] alleviates gradient conflicts via a twin-model design. (FL)^2 [38] targets few-label settings using client-specific adaptive thresholding, sharpness-aware consistency regularization, and learning-status-aware aggregation. Liu et al. [39] exploit server-side clean knowledge via representation alignment and apply shrink loss to utilize unconfident client samples safely. FedCT [40] shows that sharing hard labels on a public dataset improves the trade-off between privacy and utility. While these generic methods relax the assumption of fully labeled clients, they are not tailored to IoT anomaly detection. Crucially, they overlook

the system-level staleness and partial participation inherent to heterogeneous gateway environments.

C. Semi-Asynchronous Federated Learning

Synchronous federated learning suffers from stragglers because the server waits for the slowest client in each round, lowering efficiency in heterogeneous environments. Asynchronous federated learning performs aggregation upon update arrival [41]. However, fully asynchronous designs incur excessive communication, severe model staleness, and degraded accuracy under non-IID data. Semi-asynchronous federated learning triggers aggregation after receiving updates from a subset of clients. Early studies include SAFA [21], which tolerates bounded lag and forces synchronization for outdated clients. Similarly, FedSA [22] analyzes convergence under heterogeneous participation and designs an adaptive learning rate. Chen et al. [42] propose temporally weighted aggregation to emphasize fresh updates. Chai et al. [43] combine layered synchronization with lossy compression.

Recent methods target realistic edge scenarios. Deng et al. [44] incorporate trajectory prediction for vehicular edge computing. DP-SAFL [45] combines semi-asynchronous aggregation with differential privacy. CSAHFL [46] introduces adaptive updates for dual-layer non-IID heterogeneity in clustered settings. Although these methods improve efficiency and robustness, they assume conventional supervised learning rather than labels-at-server semi-supervised anomaly detection.

Overall, prior works advance three separate threads: federated IoT anomaly detection, label-efficient federated learning, and semi-asynchronous optimization. FedS^{3}A lies at their intersection. It targets IoT anomaly detection using unlabeled client data and limited server labels. Concurrently, it adopts a semi-asynchronous update mechanism to handle system heterogeneity and stragglers. To the best of our knowledge, the joint consideration of semi-supervised learning and semi-asynchronous aggregation remains underexplored in federated anomaly detection.

III. Preliminaries and System Architecture

This section introduces the standard federated learning procedure and formalizes the system architecture of our anomaly detection framework.

A. Federated Learning

Federated learning (FL) [16] enables distributed clients to cooperatively train a global model without exposing raw local data. This paradigm preserves data privacy while leveraging the collective knowledge of peer entities. A standard FL system consists of a centralized server S and multiple decentralized clients C_i $(1 \le i \le M)$. Clients perform local training using their private datasets. The server coordinates the process and acts as a parameter aggregator. The standard FL procedure involves the following stages:

1) Model Initialization: At round r, the server S randomly selects a subset or all of the clients C_i $(1 \le i \le N \le M)$ to participate in the training. The server then distributes the global model parameters ω_g^r to these designated clients. In the initial round, the global model parameters are initialized randomly. In subsequent rounds, the server aggregates the local model parameters uploaded by the clients to update the global model.

2) Local Training: Each participating client C_i performs E epochs of supervised training on its local dataset $D_i = \{(x_j, y_j) \mid 1 \le j \le |D_i|\}$, where x_j is the feature vector and y_j is the corresponding one-hot label. The local training objective is to minimize the empirical loss:

$$\arg\min_{\omega_i^r} \; F_i(\omega_i^r) = \frac{1}{|D_i|} \sum_{j=1}^{|D_i|} f(\omega_i^r, x_j, y_j) \tag{1}$$

where ω_i^r represents the model parameters of client C_i at round r. The function $f(\omega_i^r, x_j, y_j)$ measures the deviation between the prediction and the ground-truth label under the current parameters ω_i^r. A common example is the cross-entropy loss [47]. Optimization methods like Stochastic Gradient Descent (SGD) [48] or Adam [49] update the parameters to minimize this objective. The local update rule is given by:

$$\omega_i^{r+1} = \omega_i^r - \eta_i^r \nabla F_i(\omega_i^r) \tag{2}$$

where η_i^r is the learning rate of C_i at round r and $\nabla F_i(\omega_i^r)$ is the gradient of the loss function with respect to ω_i^r. Upon completing the local training, each client C_i uploads the updated parameters ω_i^{r+1} to the server S.

3) Global Aggregation: Upon receiving the local updates from all selected clients, the server S applies an aggregation function, such as FedAvg [16], to compute the global model for the next round. The FedAvg aggregation rule is defined as:

$$\omega_g^{r+1} = \sum_{i=1}^{N} \frac{|D_i|}{|D|} \omega_i^{r+1} \tag{3}$$

where $|D| = \sum_{i=1}^{N} |D_i|$ denotes the total number of data samples. The local datasets are mutually disjoint, meaning $D_i \cap D_{i'} = \emptyset$ for any $i \ne i'$. The global training objective is to minimize the overall loss:

$$\arg\min_{\omega_g^{r+1}} \; F_g(\omega_g^{r+1}) = \frac{\sum_{i=1}^{N} |D_i| F_i(\omega_g^{r+1})}{|D|} \tag{4}$$

This process repeats until the global model converges or the training reaches a specified maximum number of rounds R.

B. System Architecture

The IoT network model considered in this paper comprises three main entities: IoT devices, security gateways, and a security service provider. The overall system architecture is depicted in Fig. 1.

1) IoT Devices: IoT devices connect to the security gateways via protocols such as Wi-Fi, Bluetooth, or Ethernet [50]. Due to their inherent vulnerabilities, these devices are susceptible to compromise and exploitation. Therefore, the security gateway monitors their behaviors, including network communication, resource consumption, software events, and user interactions, to facilitate downstream anomaly analysis.

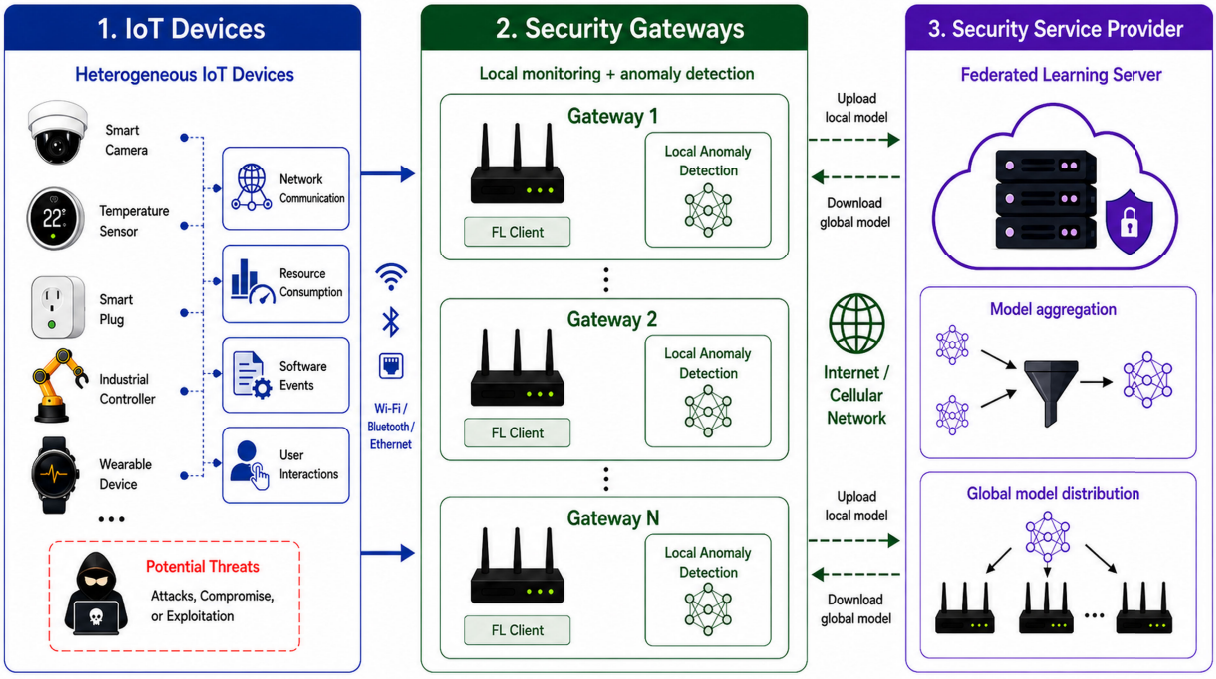


Fig. 1. System architecture of the IoT network.

2) Security Gateways: Because IoT devices have limited resources and moderate reliability, our framework does not deploy the anomaly detection component directly on them. Instead, we offload this task to the more powerful and reliable security gateways. These gateways possess sufficient resources to run robust protection mechanisms, such as Intel SGX [51] or remote attestation [52], which prevent the gateways themselves from being compromised. The gateways act as local access points to monitor the connected IoT devices and execute the trained anomaly detection models. In the FL system, they serve as the clients. They connect to the security service provider via the Internet or cellular networks to upload local updates and download the global model.

3) Security Service Provider: The security service provider acts as the central server of the FL system. This entity is typically managed by a well-resourced organization and handles the overall coordination of the federated framework. Its primary responsibilities include aggregating the local models submitted by the gateways and distributing the updated global model.

IV. FedS^{3}A Mechanism

This section details the design of FedS^{3}A, our federated semi-supervised and semi-asynchronous learning framework for anomaly detection in IoT networks.

A. Core Workflow

Fig. 2 illustrates the primary workflow of FedS^{3}A. We operate under a practical setting where clients hold entirely unlabeled data and the server maintains a small labeled dataset. Within this environment, clients perform unsupervised local training using pseudo-labeling on their extensive datasets, while the server executes supervised learning using its limited labels (Section IV-B). To balance client utilization, round efficiency, and model convergence, we introduce a semi-asynchronous update scheme (Section IV-C1). Under this scheme, the server triggers a global update immediately after receiving local models from a predefined fraction C of the clients rather than waiting for all participants. The server then aggregates these models using a specialized aggregation function (Section IV-D).

After aggregation, the server applies a staleness-tolerant distribution strategy. This strategy allows the local models of certain clients to remain asynchronous with the global model. By executing model distribution and updates for only a selected subset of clients (Section IV-C2), the framework preserves the local training progress of straggling devices and maximizes data utilization. We also adaptively adjust the learning rate of each client based on its participation frequency in global updates to balance contributions across the heterogeneous network (Section IV-E). This entire process repeats until the global model converges or the training reaches the maximum communication rounds.

B. Federated Semi-Supervised Learning

While FL provides a privacy-preserving paradigm for IoT anomaly detection, most existing schemes (Table I) rely on standard supervised frameworks. These systems assume that local client data is fully annotated with ground-truth labels and that the server acts solely as a data-free aggregator. This configuration is impractical for real-world IoT deployments. IoT devices continuously generate massive volumes of traffic data, and end users lack the expertise to annotate it. Conversely, the central server, typically managed by a security service provider, has the capability and resources to acquire a limited set of labeled data.

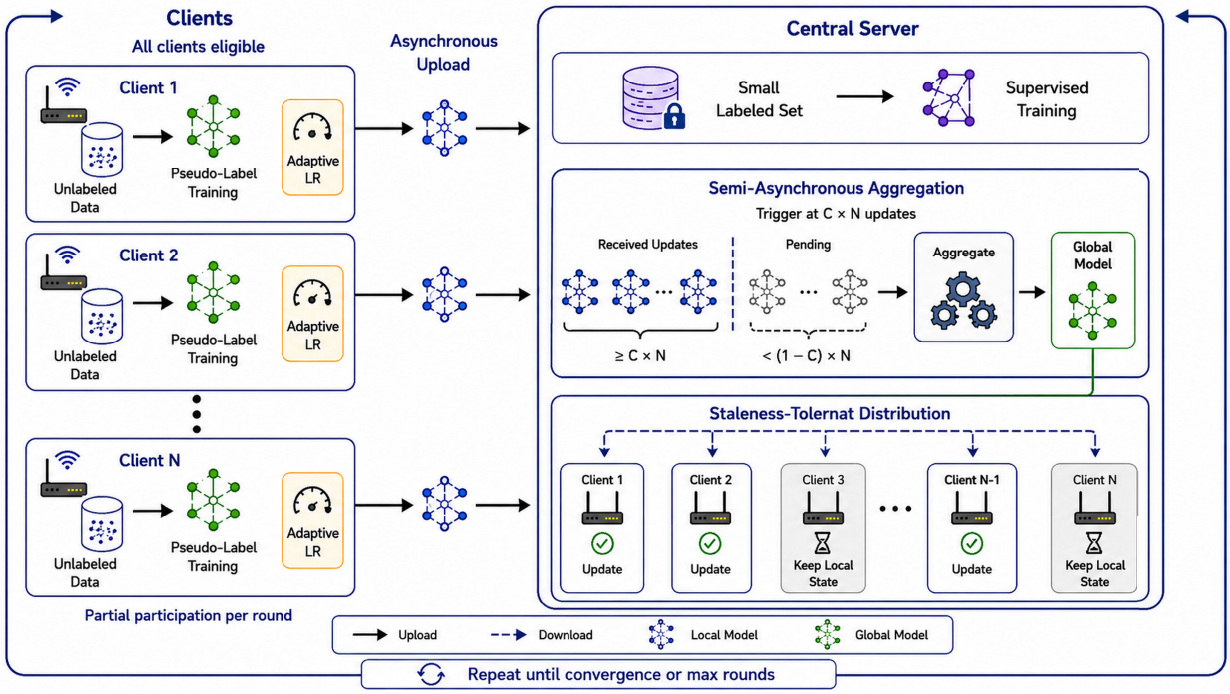


Fig. 2. The core workflow of FedS^{3}A.

TABLE I
COMPARISON OF DIFFERENT FEDERATED LEARNING APPROACHES

	Client Participation	Handling of Non-IID Data	Requirement for Training Data Labels	Communication Mode	Communication Cost Optimization
Nguyen et al. [11]	Full	No	Benign	Synchronous	None
Rey et al. [12]	Full	Yes	Yes/Benign	Synchronous	None
Campos et al. [13]	Full	Yes	Yes	Synchronous	None
Liu et al. [14]	Full	No	Yes	Synchronous	Gradient Compression
Tian et al. [15]	One	Yes	Benign	Asynchronous	Gradient Compensation
Ours	Partial	Yes	Labels at Server	Semi-Asynchronous	Sparse Matrices of Differences

Additionally, some unsupervised approaches assume that IoT devices remain completely uncompromised during the initial training stage to establish a baseline of benign behavior. This assumption is risky for resource-constrained and vulnerable IoT devices. If an adversary compromises these devices during the training phase, the system will misclassify malicious traffic as benign. This contamination significantly degrades detection accuracy and mimics the effects of data poisoning attacks [53].

To address these limitations, we formulate a disjoint Federated Semi-Supervised Learning (FSSL) scenario for IoT networks. In this setting, labeled data is exclusively available at the server, and each client holds a massive volume of unlabeled data. We detail the client and server operations below.

1) Unsupervised Learning at Clients: Because client-side samples lack annotations, we apply a pseudo-labeling method [54] to enable unsupervised local training. The client loss function is defined as follows:

$$F_i(\omega_i^r) = \frac{1}{|D_i|} \sum_{j=1}^{|D_i|} \text{sgn}(\max(\bar{y}) \geq \theta)\ l(\arg\max(\bar{y}), \bar{y}) \quad (5)$$

where $\bar{y} = p(\omega_i^r, x_j)$ represents the prediction generated by the current model parameters ω_i^r for the input sample x_j. The function l denotes the cross-entropy loss, sgn is the indicator function, and θ is a confidence threshold. We use this threshold to filter predictions and assign pseudo-labels only to high-confidence samples. This mechanism allows clients to guide their local optimization using generated pseudo-labels, effectively extracting knowledge from diverse unlabeled data distributions.

2) Supervised Learning at the Server: Relying exclusively on unlabeled client data often yields poor model convergence under non-IID conditions. To provide reliable optimization guidance, the central server leverages its proprietary labeled dataset for supervised training. During each communication round, after distributing the global model parameters to the participating clients, the server also updates these parameters locally using its labeled data. The server-side loss function is defined as follows:

$$F_s(\omega_s^r) = \frac{1}{|D_s|} \sum_{j=1}^{|D_s|} l(y_j, p(\omega_s^r, x_j)) \quad (6)$$

where ω_s^r denotes the model parameters at the server S during round r, and D_s represents the labeled dataset stored on the server.

By combining server-side supervised learning with client-side pseudo-labeling, our disjoint FSSL framework effectively exploits massive unlabeled data under strictly

limited supervision. The server's limited labeled data anchors the optimization direction and prevents the model from diverging. Concurrently, the pseudo-labeling process allows the model to learn rich traffic features from the massive volumes of client data. This dual optimization strategy mitigates the poor generalization caused by extremely limited labeled data and overcomes the low detection accuracy inherent to purely unsupervised approaches.

C. Semi-Asynchronous Federated Learning

Although synchronous FL is widely adopted for anomaly detection in IoT networks, it presents three significant limitations. First, it suffers from low client utilization. As the system pre-selects the participants for each training round, numerous capable clients remain idle even when they are ready to contribute [21]. Second, synchronous FL exhibits low round efficiency. The server must wait for all selected clients to complete local training or reach a timeout threshold before executing the aggregation. Due to the substantial heterogeneity of IoT devices, the local update times vary significantly, creating a straggler bottleneck. Third, synchronous FL leads to a waste of training progress. If selected clients fail to complete local training promptly, their computational progress is discarded when the subsequent round begins, as their local models are forcibly overwritten by the new global model.

Asynchronous FL mitigates the straggler problem but introduces new challenges. First, it incurs significant computational and communication costs because the server initiates a global update immediately upon receiving a model from any client. Second, it causes a bias in client contributions. Highly capable devices participate more frequently, thereby exerting a disproportionate impact on the global model. Third, asynchronous FL is vulnerable to staleness poisoning. Local models exhibiting severe staleness can misguide the global optimization, which decreases training accuracy, particularly under non-IID data distributions [55].

To achieve an optimal balance among client utilization, model accuracy, communication cost, and convergence rate, we design a semi-asynchronous federated learning scheme based on FSSL. This scheme comprises a semi-asynchronous model update mechanism and a staleness-tolerant distribution strategy.

1) Semi-Asynchronous Model Update: We first introduce a semi-asynchronous model update approach that uses a hyperparameter C to control the proportion of clients required to trigger a global aggregation. In conventional anomaly detection methods [11], [12], [13], [14], FedAvg [16] is the standard update rule. FedAvg pre-selects a fraction C of clients and waits for all of them to finish training. This makes it a strictly synchronous method.

In our approach, we retain the hyperparameter C but eliminate the pre-selection step. Instead, all available clients are permitted to participate simultaneously. The server triggers the model aggregation as soon as it receives local updates from a proportion C of the total clients. This design tightly couples the FL efficiency with the required participation proportion. Furthermore, synchronous and asynchronous FL can be viewed as special cases of this semi-asynchronous scheme when C approaches 1 and 0, respectively.

Due to device heterogeneity, some devices handling massive data volumes or possessing limited computational capacity will inevitably fall outside the required proportion C and fail to upload their updates in time. Nevertheless, the data residing on these slower devices is valuable. To fully utilize this knowledge and enhance model quality, we propose a staleness-tolerant distribution method.

2) Staleness-Tolerant Distribution: Version control is a critical challenge in FL. Synchronous FL simplifies this by distributing the latest model at each round, but this restricts the potential for accelerated convergence. If we deploy the semi-asynchronous update in isolation, consistently slow devices will repeatedly miss the aggregation cutoff and never participate in the FL procedure. This exclusion severely degrades the trained model's accuracy, especially in highly heterogeneous IoT networks. To fully utilize data across all clients, we introduce a staleness-tolerant distribution method to complement the semi-asynchronous update.

Our approach does not enforce strict synchronization between all clients and the server. Specifically, a subset of clients is permitted to remain asynchronous with the server. The framework tolerates outdated local models by allowing the versions on certain clients to temporarily diverge from the server's version. This encourages slower clients possessing outdated models to eventually participate in the aggregation process. Utilizing the data and computational progress of these stragglers accelerates training and improves generalization. We detail this strategy below and illustrate its integration with the semi-asynchronous update.

Based on the version gap between each client and the server, alongside the client's participation status in the global aggregation, we categorize the clients into three types:

1) Latest Clients: Clients that complete local training and successfully participate in the semi-asynchronous global model update during the current round.
2) Deprecated Clients: Clients whose local model version is excessively stale compared to the latest global model. This condition typically arises from network failures or system crashes.
3) Tolerable Clients: Clients that are not executing local training on the latest global model, yet their underlying model version is acceptably recent. This represents an intermediate state between the latest and deprecated classifications.

At round r, the server executes the model aggregation once it receives local updates from a proportion C of the clients. This yields the new global model parameters ω_g^{r+1}. Let $\omega_i^{r_i}$ represent the model parameters upon which client C_i executes local training, where r_i denotes the version of the local model for C_i. In synchronous FL, $r_i = r$ holds for all clients. In semi-asynchronous and asynchronous FL, $r_i \leq r$. The server implements the staleness-tolerant distribution strategy by executing distinct actions based on these version discrepancies.

First, for the latest clients that successfully participated in the current round, the server distributes the new global model parameters ω_g^{r+1}, regardless of their previous local

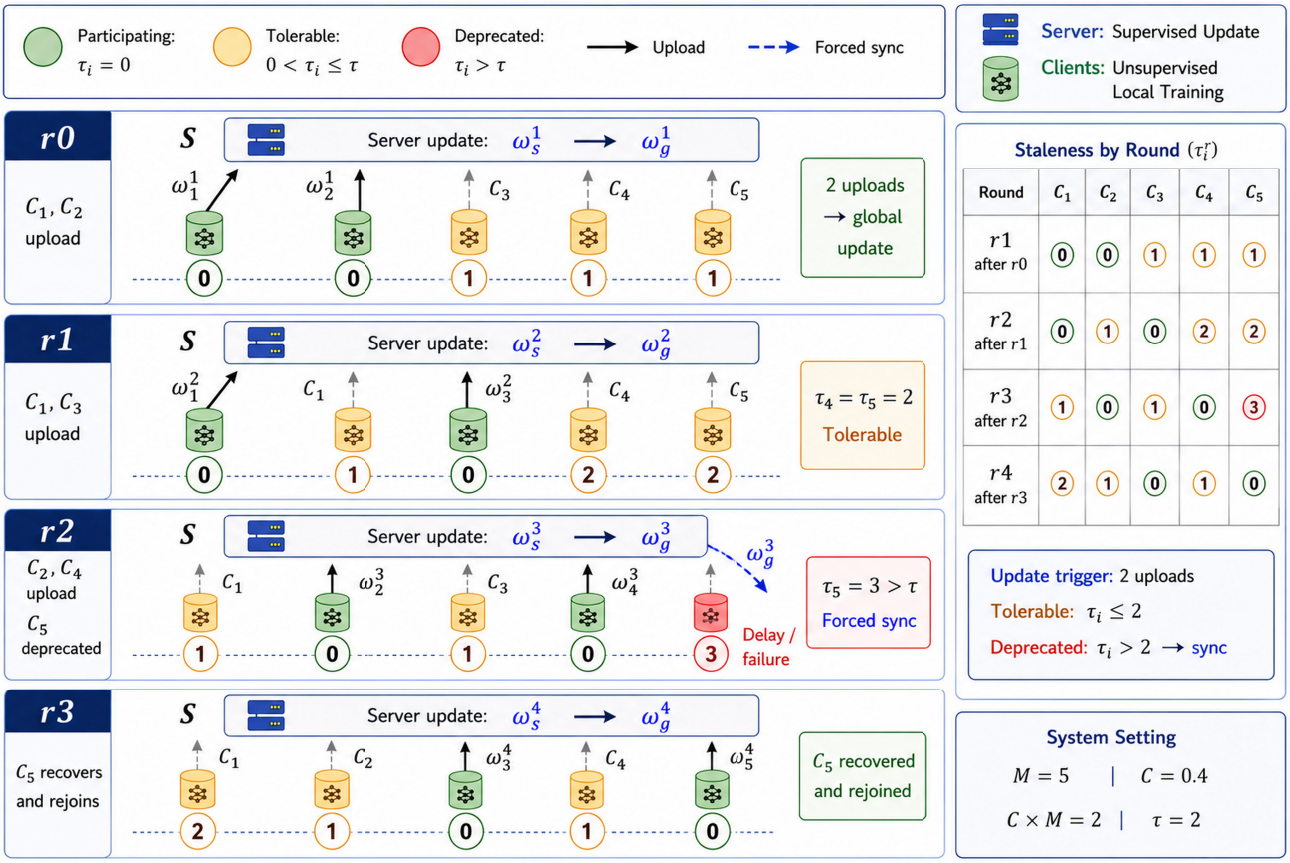


Fig. 3. Illustration of the FedS^{3}A mechanism when $C = 0.4$ and $\tau = 2$.

version. Second, for every other client C_i, the system evaluates whether the version gap between the client's resultant local model $\omega_i^{r_i+1}$ and the global model ω_g^{r+1} exceeds a predefined threshold τ. If the gap exceeds this threshold ($r - r_i > \tau$), the device is classified as a deprecated client. The server forces this client to update by distributing the latest global model to it. Otherwise, if the gap is within the limit ($r - r_i \le \tau$), the device is classified as a tolerable client. In this case, the server permits the client to remain asynchronous and refrains from forcing an update.

By combining the semi-asynchronous model update with staleness-tolerant distribution, FedS^{3}A achieves robust version control over the local models. The latest clients synchronize with the server to prevent model divergence. Deprecated clients are forced to update to ensure that severely outdated local models do not poison the global model. Meanwhile, tolerable clients can continue training asynchronously, enabling the full utilization of knowledge from slower devices. The control of model staleness is critical for guaranteeing convergence [43]. Poorly calibrated thresholds, such as an excessively small τ, cause slower clients to accumulate significant staleness before completing their local training. These clients would be continuously forced to synchronize and might never successfully participate in the global updates, severely degrading training accuracy on non-IID data.

To illustrate this mechanism, we consider the scenario in Fig. 3 comprising five clients (C_1 through C_5) and a server S. The hyperparameters are set to $C = 0.4$ and $\tau = 2$. The clients perform unsupervised learning, while the server performs supervised learning. At round r_0, clients C_1 and C_2 are the first to complete local training and upload their parameters (ω_1^1 and ω_2^1) to the server. Because the server has received parameters from $0.4 \times 5 = 2$ clients, it updates the new global model ω_g^1 using these two unsupervised updates alongside its own supervised update ω_s^1. The staleness of each client across different rounds is summarized in Fig. 3. The staleness of C_1 and C_2 is 0 due to their participation in the current round. The staleness of the remaining clients is 1. Because this is within the threshold ($\tau = 2$), they are classified as tolerable clients and are not forced to update. At round r_1, C_1 and C_3 participate in the global update. Their staleness resets to 0, while the staleness of the other clients increases by one. Next, at round r_2, C_2 and C_4 successfully complete local training. Notably, the staleness of C_5 reaches 3, which exceeds the threshold. This delay could result from network failures or system crashes. Consequently, C_5 is classified as a deprecated client, and the server distributes the latest global model ω_g^3 to force a synchronization. Finally, C_5 successfully completes training and participates in the global update at round r_3.

D. Design of Aggregation Function

Most existing FL-based anomaly detection schemes for IoT networks employ FedAvg as the aggregation function because it exhibits proven convergence properties [56]. However, our scenario involves semi-supervised learning and semi-asynchronous model updates. This setting differs from existing paradigms and requires a specialized aggregation

function. A direct application of FedAvg yields the following formulation:

$$\omega_g^{r+1} = \frac{|D_s|}{|D_c| + |D_s|}\omega_s^{r+1} + \sum_{i=1}^{CM} \frac{|D_i|}{|D_c| + |D_s|}\omega_i^{r_i+1} \tag{7}$$

where $|D_c| = \sum_{i=1}^{CM} |D_i|$. This formulation retains the weighted average principle of FedAvg while incorporating the supervised learning process at the server. However, this direct application overlooks several critical challenges specific to our scenario.

1) Revising the Weight of Supervised Learning: In practical IoT networks, unlabeled data is abundant across clients, while labeled data is scarce and restricted to the server. Despite its scarcity, this labeled data provides critical optimization guidance. Therefore, we introduce a dynamic weight for supervised learning, modeled by a function $f(r)$ that satisfies the following conditions:

1) $0 < f(r) < 1, \forall\ 0 < r \leq R$.
2) In the early stages of training, $f(r)$ should assume a large value α (e.g., $\alpha = \frac{1}{2}$). Initially, the model and the clients possess limited knowledge of the data distribution. Consequently, the supervised learning process at the server provides crucial guidance for the unsupervised learning process at the clients.
3) As the number of training rounds increases, $f(r)$ decreases monotonically and gradually approaches a constant β (i.e., $\lim_{r\to R} f(r) = \beta$). As training progresses and model performance improves, the weight of supervised learning must decrease to prevent overfitting. This decay ensures that the extensive unlabeled data at the clients is fully utilized to enhance overall performance.
4) Determining β is critical. An excessively large value leads to overfitting, while an excessively small value causes the model to forget the knowledge acquired from the labeled data. We set β to $\frac{1}{CM+1}$. This value equates the weight of the server to the average weight of the participating clients at the conclusion of training.

Incorporating this dynamic weight for supervised learning, the aggregation function is reformulated as follows:

$$\omega_g^{r+1} = f(r) * \omega_s^{r+1} + (1 - f(r)) * \sum_{i=1}^{CM} \frac{|D_i|}{|D_c|}\omega_i^{r_i+1} \tag{8}$$

2) Reducing the Weights of Stale Models: The revised aggregation function evaluates the importance of each local model based solely on the volume of training data at the corresponding client. A larger volume of training data yields a greater contribution to the global model. This principle is standard in synchronous FL. However, our framework employs a semi-asynchronous update mechanism and a staleness-tolerant distribution scheme. At round r, the model version used by client C_i for local training may not be current. Consequently, r_i does not necessarily equal r. Because FL performance relies heavily on local model quality, increased staleness can introduce substantial errors during global aggregation.

Weighting all participating local models solely by data volume, irrespective of their model versions, is suboptimal. Clients operating on outdated model versions should receive discounted importance. To address this, we optimize the aggregation function to incorporate local model staleness. A higher degree of staleness yields a lower aggregation weight. The staleness-aware aggregation function is defined as follows:

$$\omega_g^{r+1} = f(r)\,\omega_s^{r+1} + \big(1 - f(r)\big) \sum_{i=1}^{K} \frac{|D_i|}{|D_c|}\, g(r - r_i)\,\omega_i^{r_i+1} \tag{9}$$

where $g(r - r_i)$ represents the staleness function. This function equals 1 when $r_i = r$ and decreases monotonically as $r - r_i$ increases. Various functions satisfy these properties with different decay rates (e.g., $g(r - r_i) = a^{-(r-r_i)}$ for $a > 1$).

3) Group-Based Aggregation Function: The non-IID nature of the local data can induce substantial variation in the models optimized by different clients. This variation destabilizes the global aggregation and decelerates convergence [57]. To mitigate this issue, we extend the staleness-aware aggregation with a group-based strategy that accounts for data distribution heterogeneity. Upon receiving updates during a semi-asynchronous round, the server applies K-means clustering to partition the participating clients into $|G|$ groups based on their data distribution statistics. The server then performs hierarchical aggregation across these groups. Building upon the staleness-aware aggregation presented in Eq. (9), the global model is updated as follows:

$$\begin{cases} \omega_g^{r+1} = f(r)\,\omega_s^{r+1} + \big(1 - f(r)\big)\dfrac{\sum_{k=1}^{|G|} \omega_{G_k}^{r+1}}{|G|}, \\ \omega_{G_k}^{r+1} = \displaystyle\sum_{C_i \in G_k} \frac{|D_i|}{|D_{G_k}|}\, g(r - r_i)\,\omega_i^{r_i+1}, \end{cases} \tag{10}$$

where $|D_{G_k}| = \sum_{C_i \in G_k} |D_i|$. Within each group, the local models are aggregated using data-volume weights combined with the staleness attenuation function $g(r - r_i)$. The server then uniformly averages the group-level models. This ensures that groups with diverse data distributions contribute equally to the unsupervised component of the global update. This design prevents high-volume groups from dominating the aggregation process, helping the global model accurately capture the heterogeneous data distributions.

This grouping mechanism addresses statistical heterogeneity under the standard FL assumption that clients report their data distribution statistics honestly. It does not account for adversarial manipulation of these statistics. Therefore, this design functions as a heterogeneity-aware aggregation strategy rather than an adversary-resilient defense. We reserve the incorporation of robust or trust-aware grouping methods against adversarial environments for future work.

E. Adaptive Learning Rate

The semi-asynchronous model update and staleness-tolerant distribution schemes introduce diversity in local model versions. Furthermore, device heterogeneity causes variations in client participation frequencies during global aggregation. Let f_i denote the relative frequency with which client C_i participates in the global update, such that $\sum_{i=1}^{M} f_i = 1$. We apply an adaptive learning rate η_i to each client based on

its relative frequency to balance contributions and enhance learning performance.

Intuitively, if a client participates in global updates more frequently because of a smaller data volume, greater computational capacity, or faster network speeds, its learning rate should decrease. Conversely, the learning rate of infrequent participants should increase to allow the global model to extract sufficient information from their local data. The adaptive learning rate η_i for client C_i is defined as follows:

$$\eta_i = \frac{\lambda}{M * f_i} \tag{11}$$

where λ represents the global base learning rate, M indicates the total number of clients, and f_i denotes the relative frequency of client C_i. A direct approach to measuring this frequency is to count the historical participation rounds of each client and compute a simple weighted average:

$$f_i = \frac{n_i}{\sum_{j=1}^{M} n_j} \tag{12}$$

where n_i represents the number of times client C_i has participated in the global update. While intuitive, this approach treats all past rounds equally and assigns identical weights. This overlooks a critical temporal factor.

Consider the scenario in Fig. 3, where client C_1 participates at rounds r_0 and r_1, C_2 participates at r_0 and r_2, and C_3 participates at r_1 and r_3. All three clients participate in exactly two rounds. According to the simple average formula, their frequencies are evaluated equally. However, the influence of a local model uploaded at an earlier round r' on the current global model gradually decays as the training progresses to round r. The global model progressively overwrites information derived from older local models and prioritizes recently acquired knowledge.

Consequently, recent participation exerts a greater impact on the global model. The relative impact of clients C_3, C_2, and C_1 on the current global model should be evaluated in descending order ($C_3 > C_2 > C_1$). Therefore, recent rounds must receive higher weights, while older rounds receive lower weights. We employ a round-weight function inspired by exponential smoothing [58] to satisfy this requirement, defined as $h(r) = (1 + a)^r$ for $a > 0$.

F. Communication Cost Optimization

In federated anomaly detection, communication costs are primarily determined by the volume of model parameters exchanged per round and the total number of rounds required for convergence. On the one hand, increasing local training epochs can reduce the number of communication rounds. However, an increase in local epochs tends to amplify local model divergence, particularly under non-IID data distributions. On the other hand, techniques like structured and sketched updates [59] reduce the transmitted data volume per round. Unfortunately, many compression methods inevitably introduce parameter errors [60], [61]. The aggregation of these distorted models can reduce accuracy and decelerate convergence.

To minimize communication costs while preserving model quality and convergence speed, we propose a sparse difference transmission approach. Specifically, we apply L1 regularization to the model parameters to induce sparsity. Upon completing local training, client C_i computes the difference between its updated local parameters and the initial global parameters received for that round ($\Delta\omega = \omega_i^{r_i+1} - \omega_g^{r_i}$). This step isolates the knowledge acquired during the current round. The client then transmits this difference as a sparse matrix to optimize bandwidth utilization. Similarly, following the global update, the server transmits only the sparse difference matrix to the clients requiring an update.

G. FedS^{3}A Overview

Algorithm 1 summarizes the complete execution flow of FedS^{3}A. At a high level, the framework alternates among supervised updating at the server, local training via pseudo-labeling at the clients, partial asynchronous aggregation, and staleness-aware redistribution. The detailed steps are outlined as follows:

1) Model Initialization and Server Warm-up: At the onset of training, the server S initializes the global model and conducts supervised learning on the labeled dataset D_s for E_s epochs. The resulting model ω_g^0 is distributed to all clients C_i ($1 \le i \le M$).
2) Local Update at the Clients: Upon receiving the global model, each client C_i executes E epochs of local training on its unlabeled dataset D_i using pseudo-labels. The client then uploads the updated local model $\omega_i^{r_i+1}$ to the server. Due to heterogeneous data volumes, computational capacities, network conditions, and potential failures, different clients may return updates derived from different global model versions.
3) Partial Asynchronous Aggregation: During each global round, the server first executes a supervised update on D_s. It then collects updates from the clients asynchronously. Once the server receives updates from a predefined fraction C of the clients, it partitions the participating clients into groups based on their data distribution statistics and performs group-based global aggregation using Eq. (10).
4) Adaptive Redistribution: After aggregation, the server updates the participation frequency estimates, computes the adaptive learning rates, and redistributes the new global model ω_g^{r+1}. This model is sent only to the participating clients and to those holding local models that exceed the staleness tolerance τ.
5) Iterative Optimization: Steps 2 through 4 are repeated until the maximum number of rounds R is reached.

In Algorithm 1, f_i denotes the estimated participation frequency of client C_i for adjusting the adaptive learning rate, and v_i represents the latest version of the global model currently held by C_i for staleness control.

V. Experimental Results

This section evaluates the performance of the FedS^{3}A framework for anomaly detection within IoT networks.

Algorithm 1 FedS^3 A Training Procedure

Input: Server labeled dataset D_s; client unlabeled datasets $\{D_i\}_{i=1}^{M}$; number of clients M; client participation ratio C; total number of global rounds R; server warm-up epochs E_s; local training epochs E; staleness tolerance τ; number of groups $|G|$; base learning rate λ

Output: Trained global model ω_g^R

Initialize the global model ω_g^0
$\omega_g^0 \leftarrow \text{SUPERVISEDUPDATE}\omega_g^0, D_s, E_s$
for $i = 1$ to M **do**
  $f_i \leftarrow 1/M$, $v_i \leftarrow 0$, $\eta_i^0 \leftarrow \lambda$
  Send (ω_g^0, η_i^0) to client C_i
end for
for $r = 0$ to $R - 1$ **do**
  $\omega_s^{r+1} \leftarrow \text{SUPERVISEDUPDATE}\omega_g^r, D_s, E$
  $K \leftarrow \lceil CM \rceil$, $V \leftarrow \emptyset$, $\mathcal{U}_r \leftarrow \emptyset$
  while $|V| < K$ **do**
    Receive $(i, \omega_i^{r_i+1}, r_i)$ from some client C_i
    if $i \notin V$ **then**
      $V \leftarrow V \cup \{i\}$
      $\mathcal{U}_r \leftarrow \mathcal{U}_r \cup \{(i, \omega_i^{r_i+1}, r_i)\}$
    end if
  end while
  Partition the participating clients in V into $|G|$ groups by K-means on their data-distribution statistics
  $\omega_g^{r+1} \leftarrow \text{AGGREGATE}\omega_s^{r+1}, \mathcal{U}_r, r$ ▷ Eq. (10)
  Update the participation-frequency estimates $\{f_i\}_{i=1}^{M}$
  for $i = 1$ to M **do**
    $\eta_i^{r+1} \leftarrow \lambda/(Mf_i)$ ▷ Eq. (11)
    if $i \in V$ **or** $(r + 1) - v_i > \tau$ **then**
      Send $(\omega_g^{r+1}, \eta_i^{r+1})$ to client C_i
      $v_i \leftarrow r + 1$
    end if
  end for
end for
return ω_g^R
procedure ONRECEIVE$(i, \omega_g^r, \eta_i^r)$
  Generate pseudo-labels for the high-confidence samples in D_i
  $\omega_i^{r+1} \leftarrow \text{LOCALUPDATE}\omega_g^r, D_i, E, \eta_i^r$
  Upload (i, ω_i^{r+1}, r) to the server
end procedure
procedure SUPERVISEDUPDATE(ω, D_s, E)
  for $e = 1$ to E **do**
    Update ω by minimizing the supervised objective on D_s
  end for
  return ω
end procedure
procedure LOCALUPDATE(ω, D_i, E, η_i)
  for $e = 1$ to E **do**
    Update ω by minimizing the pseudo-label objective on D_i
  end for
  return ω
end procedure

TABLE II
SUMMARY OF DATASETS

Dataset	Feature Dim.	Attack Types	Subset Size	Train/Test	Server Labels
CIC-IDS2017	78	8	~540,000	9:1	5%
Edge-IIoTset	61	12	155,585	9:1	5%

A. Datasets and Scenarios

Table II summarizes the datasets and the default experimental settings. We evaluate FedS^{3}A on two public intrusion detection benchmarks: CIC-IDS2017 [27] and Edge-IIoTset [28]. CIC-IDS2017 serves as a standard benchmark for network intrusion detection. Edge-IIoTset is a recent IIoT-oriented benchmark designed for both centralized and federated learning architectures.

For CIC-IDS2017, we utilize a processed version containing 78-dimensional feature vectors and 15 traffic classes. Following common practice, we remove attack categories with extremely sparse samples and retain eight attack types for evaluation: DoS Hulk, PortScan, DDoS, DoS GoldenEye, FTP-Patator, SSH-Patator, DoS slowloris, and DoS Slowhttp. Because benign traffic dominates the original dataset, we randomly sample a subset of approximately 540,000 records to mitigate severe class imbalance.

For Edge-IIoTset, we use the processed feature dataset comprising 61 selected features and 15 traffic classes. Due to the highly imbalanced class distribution and the extremely limited samples in specific attack categories, we construct a compact evaluation subset by discarding the Fingerprinting and MITM categories. We retain 12 attack types: Backdoor, DDoS_HTTP, DDoS_ICMP, DDoS_TCP, DDoS_UDP, Password, Port_Scanning, Ransomware, SQL_injection, Uploading, Vulnerability_scanner, and XSS. The resulting subset contains 155,585 samples.

For both datasets, we split the training and test sets at a ratio of 9:1. To replicate the disjoint FSSL setting, the server holds exactly 5% of the training data along with their ground-truth labels. The clients hold the remaining 95% of the training data entirely unlabeled.

To evaluate the framework under varying degrees of statistical heterogeneity, we define a Basic scenario and a Balanced scenario. In the Basic scenario, both the class proportions and the local data volumes vary substantially across the clients. This setup reflects practical IoT deployments where different gateways observe traffic from distinct device populations and experience different attack exposures. In the Balanced scenario, the class proportions are roughly uniform across the clients, although local data volumes may still differ. Unless otherwise stated, the default federation consists of 10 clients. Both datasets are partitioned according to these scenario principles.

B. Experiment Setup

We implement the experiments using the Keras library with a TensorFlow backend [62]. The simulated federated system comprises one central server and 10 clients acting as security gateways. Each gateway monitors multiple heterogeneous IoT devices and runs a local anomaly detection model. We employ

TABLE III
DEFAULT SIMULATION SETTINGS

	Basic	*Balanced*
Staleness Function	Exponential	Polynomial
Round-weight Function	Exponential	Exponential Smoothing
Staleness Tolerance	2	
Participation Proportion of Clients	0.6	
Optimizer	Adam	
Learning Rate	0.0001	
Batch Size	100	
Epoch	1	
Pseudo-labeling Threshold	0.95	

a Convolutional Neural Network (CNN) for the detection task. The model consists of two 1D-CNN layers (with 128 and 256 filters, respectively), one flatten layer, one fully-connected layer (with 256 hidden neurons and ReLU activation), one dropout layer (with a dropout rate of 0.1), and a final fully-connected layer using Softmax activation. Table III lists the remaining default experimental settings.

We simulate the FL environment on a workstation equipped with an 80-core Intel Xeon Gold 6230N CPU running at 2.30 GHz and 352 GB of RAM. Although we conduct the simulations on a single physical machine, we isolate the federated entities into independent operating system processes. Each process acts exclusively as either the server or a distinct client to execute the federated training in parallel.

C. Evaluation Metrics

We categorize our evaluation metrics into detection metrics and federated system metrics. Packet-level network QoS metrics, such as end-to-end forwarding delay and throughput in live deployments, fall outside the scope of this evaluation.

1) Detection Metrics: We utilize five standard metrics to evaluate the anomaly detection performance: $Accuracy = \frac{TP+TN}{TP+TN+FP+FN}$, $Precision = \frac{TP}{TP+FP}$, $Recall = \text{TPR} = \frac{TP}{TP+FN}$, $F1\text{-}score = \frac{2TP}{2TP+FP+FN}$, and $FPR = \frac{FP}{FP+TN}$, where TP, TN, FP, and FN denote true positives, true negatives, false positives, and false negatives, respectively. Because the evaluation subsets for both CIC-IDS2017 and Edge-IIoTset are multi-class and imbalanced, we report the support-weighted averages of these metrics. Specifically, we compute each metric per class in a one-vs-rest manner and calculate the final result by averaging across classes with weights proportional to their sample counts. The F1 score and FPR are particularly informative because they jointly reflect the detection capability and false-alarm rates under imbalanced distributions.

2) Federated System Metrics: To evaluate the efficiency of FL across heterogeneous clients, we utilize two system-level metrics:

1) Average Communication Overhead (ACO): the average ratio per global round of the total data volume communicated between the server and the participating clients to the total size of the model parameters.
2) Average Round Time (ART): the average wall-clock time of a global round. This duration is measured from the moment the server distributes the global model until the corresponding aggregation is completed.

TABLE IV
RESULTS SUMMARY OF DIFFERENT STALENESS FUNCTIONS

Dataset	Scenario	Function	Acc.	Prec.	Recall	F1	FPR
CIC-IDS2017	Basic	Constant	0.9708	0.9708	0.8489	0.8983	0.0071
		Polynomial	0.9780	0.9780	0.8586	0.9071	**0.0022**
		Hinge	0.9785	0.9785	0.8597	0.9065	0.0063
		Exponential	**0.9818**	**0.9818**	**0.8965**	**0.9312**	0.0059
	Balanced	Constant	0.9688	0.9687	0.8241	0.8799	0.0083
		Polynomial	**0.9815**	**0.9815**	**0.8684**	**0.9130**	**0.0032**
		Hinge	0.9781	0.9781	0.8424	0.8930	0.0082
		Exponential	0.9810	0.9810	0.8669	0.9119	0.0034
Edge-IIoTset	Basic	Constant	0.9794	0.9794	0.8928	0.9331	0.0053
		Polynomial	0.9847	0.9847	0.9108	0.9439	**0.0031**
		Hinge	0.9850	0.9850	0.9121	0.9436	0.0046
		Exponential	**0.9864**	**0.9864**	**0.9231**	**0.9486**	0.0040
	Balanced	Constant	0.9821	0.9821	0.9041	0.9400	0.0050
		Polynomial	**0.9874**	**0.9874**	**0.9301**	**0.9525**	**0.0030**
		Hinge	0.9859	0.9859	0.9195	0.9466	0.0044
		Exponential	0.9872	0.9872	0.9288	0.9520	0.0033

The ACO characterizes the communication burden of the federated training process. The ART reflects the round-level training efficiency under client heterogeneity and semi-asynchronous updates. Unless otherwise stated, we independently repeat each experiment 10 times and report the mean values as the final results.

D. Parameter Selection

1) Impact of Different Staleness Functions: We evaluate four functions. The constant function (g_1) represents the baseline with no adaptive decay for stale local updates. For the remaining decay-based functions, we configure the parameters as follows: $a = \frac{1}{2}$ for the polynomial function g_2, $a = 1$ and $b = 0$ for the hinge function g_3, and $a = \frac{e}{2}$ for the exponential function g_4.

As shown in Table IV, the three decay-based staleness functions (g_2 through g_4) consistently outperform the constant function across all datasets and scenarios. This indicates that explicitly accounting for update staleness benefits the semi-asynchronous setting. An appropriate staleness decay mitigates the adverse effects of outdated local updates on the global model while preserving the valuable knowledge extracted from slower clients.

For both the CIC-IDS2017 and Edge-IIoTset datasets, the exponential function achieves the highest overall performance in the Basic scenario. The polynomial function performs best in the Balanced scenario, although the exponential function maintains highly competitive accuracy. This consistent trend across both datasets demonstrates that the optimal staleness function depends on the data distribution scenario rather than the specific dataset. Consequently, we adopt the exponential staleness function for the Basic scenario and the polynomial staleness function for the Balanced scenario as the default configurations in all subsequent experiments.

2) Impact of Different Round-Weight Functions: We investigate the impact of the adaptive learning rate alongside five distinct round-weight functions:

- Constant:$h_1(r) = 1$
- Logarithmic:$h_2(r) = \ln(1 + r)$
- Polynomial:$h_3(r) = (1 + r)^a$
- Exponential Smoothing:$h_4(r) = (1 + a)^r$
- Exponential:$h_5(r) = a^r$

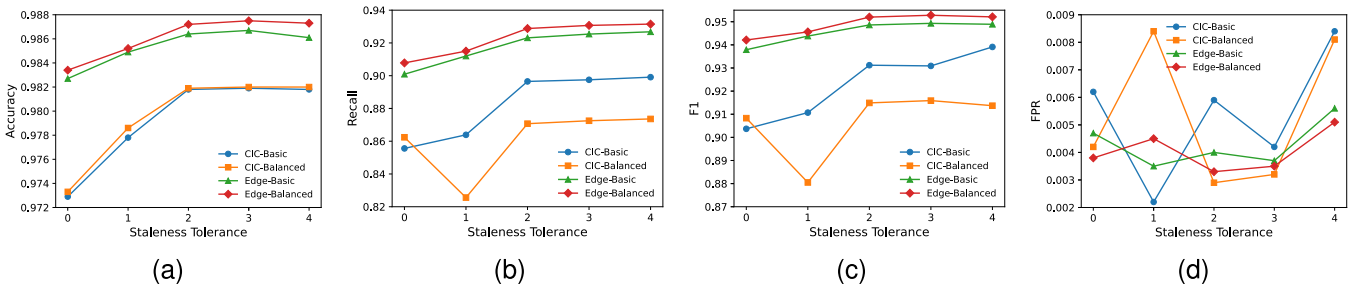


Fig. 4. The impact of staleness tolerance on detection performance. (a) Accuracy; (b) recall; (c) F1 score; and (d) FPR.

We set the parameters as follows: $a = \frac{1}{2}$ for the polynomial function $h_3(r)$, $a = 0.1$ for the exponential smoothing function $h_4(r)$, and $a = \frac{e}{2}$ for the exponential function $h_5(r)$. As shown in Table V, the four monotonically increasing round-weight functions (h_2 through h_5) consistently outperform the constant baseline h_1 across all datasets and scenarios. Assigning larger weights to more recent rounds aligns the client updates closer to the current state of the global model, thereby enhancing aggregation effectiveness in the semi-asynchronous setting.

For both datasets, the exponential function achieves the highest overall performance in the Basic scenario. The exponential smoothing function performs best in the Balanced scenario. This stable trend confirms that the optimal round-weight function is not dataset-specific. Thus, we adopt the exponential function for the Basic scenario and the exponential smoothing function for the Balanced scenario as default configurations.

We also conduct an ablation study to isolate the effect of the adaptive learning rate. Models employing the adaptive learning rate consistently outperform non-adaptive configurations across all datasets, scenarios, and round-weight functions. Device heterogeneity naturally causes uneven client participation. Applying a uniform learning rate biases the global model toward clients with higher participation frequencies. This bias degrades overall detection performance. The adaptive learning rate effectively counteracts this imbalance.

3) Impact of Different Staleness Tolerances: Fig. 4 presents the performance of FedS^{3}A under different staleness tolerance values. Across all datasets and scenarios, a staleness tolerance of 0 generally yields lower Accuracy, Recall, and F1 scores compared to configurations with moderate tolerance values. Because FedS^{3}A initiates global aggregation after receiving updates from a subset of gateways, a zero tolerance forces the remaining slower gateways to interrupt local training and synchronize excessively. This frequent synchronization diminishes the utilization of local data on these slower devices.

As the tolerance increases from 0 to 2, the Accuracy, Recall, and F1 scores improve steadily across most configurations. When the tolerance increases to 3 or 4, the overall performance stabilizes and additional gains become marginal. However, the FPR fluctuates under higher tolerances. This indicates that permitting overly stale local models to participate in the aggregation process destabilizes the system's false-positive behavior.

These results suggest that a moderate staleness tolerance provides an effective balance between accommodating asynchronous updates and mitigating the adverse effects of stale local models. We adopt a value of 2 as the default staleness tolerance in subsequent experiments.

TABLE V
RESULTS SUMMARY OF DIFFERENT ROUND-WEIGHT FUNCTIONS

Dataset	Scenario	Function	Acc.	Prec.	Recall	F1	FPR
CIC-IDS2017	Basic	Non-Adaptive	0.9678	0.9678	0.8404	0.8909	0.0090
		Constant	0.9698	0.9698	0.8444	0.8952	0.0068
		Logarithmic	0.9712	0.9712	0.8545	0.9023	0.0051
		Polynomial	0.9739	0.9739	0.8599	0.9076	0.0027
		Exp. Smoothing	0.9737	0.9737	0.8578	0.9066	**0.0021**
		Exponential	**0.9818**	**0.9818**	**0.8965**	**0.9312**	0.0059
	Balanced	Non-Adaptive	0.9568	0.9568	0.8028	0.8630	0.0104
		Constant	0.9592	0.9592	0.8088	0.8683	0.0094
		Logarithmic	0.9734	0.9734	0.8574	0.9056	0.0032
		Polynomial	0.9752	0.9752	0.8685	0.9130	0.0033
		Exp. Smoothing	**0.9819**	**0.9819**	**0.8707**	**0.9149**	**0.0029**
		Exponential	0.9815	0.9815	0.8684	0.9130	0.0032
Edge-IIoTset	Basic	Non-Adaptive	0.9801	0.9801	0.8956	0.9357	0.0056
		Constant	0.9810	0.9810	0.8994	0.9376	0.0049
		Logarithmic	0.9830	0.9830	0.9076	0.9419	0.0039
		Polynomial	0.9848	0.9848	0.9148	0.9454	0.0032
		Exp. Smoothing	0.9846	0.9846	0.9131	0.9446	**0.0029**
		Exponential	**0.9864**	**0.9864**	**0.9231**	**0.9486**	0.0040
	Balanced	Non-Adaptive	0.9792	0.9792	0.8921	0.9330	0.0058
		Constant	0.9804	0.9804	0.8968	0.9356	0.0050
		Logarithmic	0.9843	0.9843	0.9145	0.9459	0.0038
		Polynomial	0.9857	0.9857	0.9223	0.9496	0.0037
		Exp. Smoothing	**0.9872**	**0.9872**	**0.9288**	**0.9520**	**0.0033**
		Exponential	0.9870	0.9870	0.9274	0.9512	0.0035

4) Impact of Different Participation Proportions: We evaluate the effect of different participation proportions on model performance and round efficiency. We test participation proportions of 0.1, 0.4, 0.5, 0.6, and 1.0. A proportion of 0.1 corresponds to a highly asynchronous configuration, while 1.0 indicates a fully synchronous setting.

As shown in Fig. 5, across all datasets and scenarios, a participation proportion of 0.1 generally yields the lowest Accuracy and F1 scores alongside a relatively high FPR. Under this setting, the server initiates global aggregation after receiving updates from only a minimal fraction of gateways. This premature aggregation exacerbates the staleness of the remaining local models. Although FedS^{3}A incorporates an adaptive staleness function, excessively asynchronous aggregation still degrades the final model quality.

As the participation proportion increases, the Accuracy and F1 scores improve. The server incorporates updates from a larger number of gateways and utilizes local data more effectively in each round. However, this improvement degrades round efficiency. As demonstrated in Fig. 5(d), the ART increases monotonically with the participation proportion on both datasets. The server must wait for additional gateways to complete local training before executing the global aggregation.

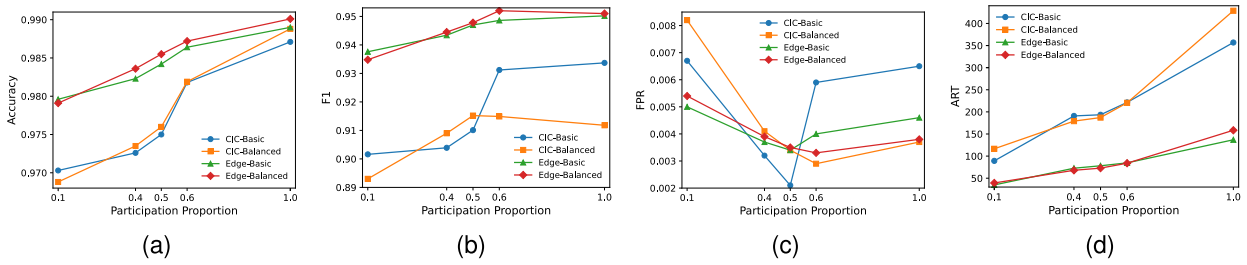


Fig. 5. The impact of participation proportion on detection performance and round efficiency. (a) Accuracy; (b) F1 score; (c) FPR; and (d) ART.

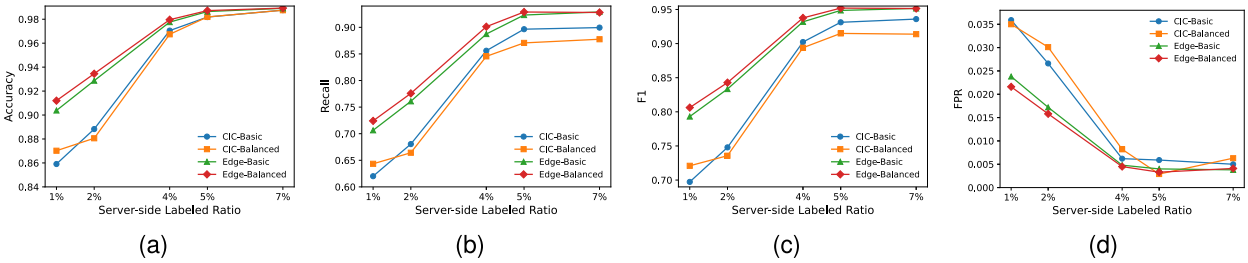


Fig. 6. The impact of the server-side labeled ratio on detection performance. (a) Accuracy; (b) Recall; (c) F1 score; and (d) FPR.

Although the fully synchronous setting (a proportion of 1.0) achieves the highest Accuracy, the marginal gains in the F1 score over the 0.6 setting remain limited and inconsistent. Additionally, the FPR fails to improve monotonically. The ART increases substantially when the participation proportion rises from 0.6 to 1.0. To attain an optimal trade-off between detection performance and round efficiency, we adopt 0.6 as the default participation proportion.

5) Impact of Different Data Sizes on the Server: We evaluate how the amount of server-side labeled data impacts model performance. In our disjoint federated semi-supervised learning setting, only the server holds labeled data. We vary the ratio of server-side labeled data relative to the total training data across 1%, 2%, 4%, 5%, and 7%.

As shwon in Fig. 6, even when the labeled data ratio is merely 1%, FedS^{3}A achieves strong detection performance across both datasets. The framework effectively exploits the massive unlabeled gateway data under strictly limited supervision. As the labeled data ratio increases, the Accuracy, Recall, and F1 scores improve substantially, and the FPR decreases markedly. A larger volume of labeled data provides stronger supervision for the global model, enabling it to generate more accurate pseudo-labels at the gateways.

When the ratio increases from 2% to 4%, the gains in Accuracy, Recall, and F1 score are pronounced across all scenarios, accompanied by a substantial reduction in the FPR. However, when the ratio further increases from 5% to 7%, additional gains become marginal. Several metrics exhibit slight fluctuations rather than consistent improvements. This suggests that the model performance approaches a stable regime at a ratio of approximately 5%.

To achieve an optimal balance between detection performance and labeling cost, we adopt 5% as the default labeled data ratio in subsequent experiments.

E. Ablation Study

1) Impact of the Group-Based Aggregation Function: We evaluate the group-based aggregation function exclusively in the Basic scenario across both datasets, as this setting exhibits strong non-IID characteristics. Because the aggregation mechanism clusters clients based on their data distributions, grouping provides minimal benefits in the Balanced scenario where client data is approximately IID.

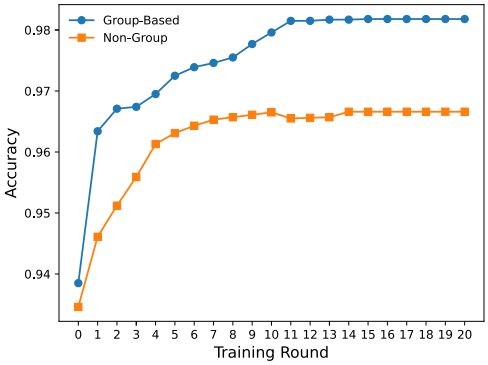


Fig. 7. The impact of group-based aggregation function.

TABLE VI
IMPACT OF THE GROUP-BASED AGGREGATION FUNCTION UNDER THE BASIC SCENARIO

Dataset	Method	Acc.	Prec.	Recall	F1	FPR
CIC-IDS2017	Non-Group	0.9666	0.9666	0.8357	0.8886	**0.0054**
	Group-Based	**0.9818**	**0.9818**	**0.8965**	**0.9312**	0.0059
Edge-IIoTset	Non-Group	0.9780	0.9780	0.8898	0.9318	**0.0036**
	Group-Based	**0.9864**	**0.9864**	**0.9231**	**0.9486**	0.0040

Table VI and Fig. 7 show that group-based aggregation consistently enhances overall model performance on both the CIC-IDS2017 and Edge-IIoTset datasets under the Basic scenario. Compared to the Non-Group baseline, the Group-Based variant achieves pronounced gains in Accuracy, Recall, and F1 scores. Grouping clients by data distribution enables FedS^{3}A to capture representative gradient information from distinct client clusters. This approach successfully alleviates the adverse effects of non-IID data on global model convergence.

2) Impact of the Dynamic Weight of Supervised Learning: To verify the necessity of a dynamic weight for supervised

TABLE VII
IMPACT OF THE DYNAMIC WEIGHT OF SUPERVISED LEARNING

Dataset	Scenario	Method	Acc.	Prec.	Recall	F1	FPR
CIC-IDS2017	Basic	Static-$\frac{1}{2}$	0.9703	0.9703	0.8517	0.9000	**0.0055**
		Adaptive	**0.9818**	**0.9818**	**0.8965**	**0.9312**	0.0059
		Static-$\frac{1}{7}$	0.9688	0.9688	0.8471	0.8963	0.0061
	Balanced	Static-$\frac{1}{2}$	0.9708	0.9708	0.8494	0.8991	0.0047
		Adaptive	**0.9819**	**0.9819**	**0.8707**	**0.9149**	**0.0029**
		Static-$\frac{1}{7}$	0.9727	0.9727	0.8551	0.9038	0.0036
Edge-IIoTset	Basic	Static-$\frac{1}{2}$	0.9803	0.9803	0.8997	0.9383	**0.0038**
		Adaptive	**0.9864**	**0.9864**	**0.9231**	**0.9486**	0.0040
		Static-$\frac{1}{7}$	0.9791	0.9791	0.8948	0.9352	0.0043
	Balanced	Static-$\frac{1}{2}$	0.9808	0.9808	0.9053	0.9413	0.0042
		Adaptive	**0.9872**	**0.9872**	**0.9288**	**0.9520**	**0.0033**
		Static-$\frac{1}{7}$	0.9821	0.9821	0.9108	0.9449	0.0037

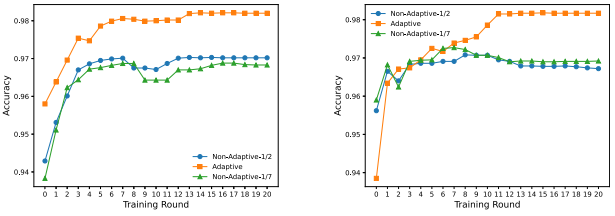


Fig. 8. The impact of the dynamic weight of supervised learning on detection accuracy. (a) Basic Scenario. (b) Balanced Scenario.

learning, we compare it against static weighting strategies. As defined previously, our dynamic weight decays from $\frac{1}{2}$ to $\frac{1}{CM+1}$ (which equals $\frac{1}{7}$ given $C = 0.6$ and $M = 10$) as training progresses. Table VII reports the model performance when this weight remains fixed at either $\frac{1}{2}$ or $\frac{1}{7}$.

The dynamically decaying weight consistently outperforms both fixed-weight configurations across all datasets and scenarios (Table VII). The dynamic strategy provides strong supervision during the early stages of training to guide the generation of accurate pseudo-labels at the clients. By gradually reducing the supervised contribution in later rounds, the system prevents overfitting to the limited server-side labeled data. Thus, FedS^{3}A achieves an optimal balance between utilizing server supervision and exploiting large-scale unlabeled client data.

Fig. 8 reveals divergent trends for the fixed weights depending on the scenario. The model employing a fixed weight of $\frac{1}{2}$ outperforms the $\frac{1}{7}$ configuration in the Basic scenario, but the opposite occurs in the Balanced scenario. Under the highly non-IID distributions of the Basic scenario, an excessively small fixed weight fails to provide sufficient server guidance for the unsupervised clients. In the Balanced scenario, an excessively large fixed weight forces the global model to overfit the server's labeled data. This over-reliance limits the model's ability to learn from the massive unlabeled client datasets. The dynamic weight elegantly navigates both extremes, justifying its inclusion in the FedS^{3}A framework.

F. Performance Comparison

We compare FedS^{3}A against representative baselines across three categories: (i) classical federated learning controls, (ii) recent semi-supervised federated learning methods with server-side labels, and (iii) an application-specific method for IoT federated intrusion detection. Because these methods rely on distinct assumptions, optimization pipelines, and communication paradigms, reproducing them in a perfectly identical form is unfeasible. We preserve the core design of each baseline while aligning the data partitions, model backbones, and evaluation protocols to ensure a fair and informative comparison.

1) Benchmarks:

a) **FedAvg-SSL:** FedAvg [16] is a mainstream federated optimization method widely adopted for anomaly detection in IoT networks [11], [12], [13], [14]. We evaluate two variants based on gateway participation: FedAvg-SSL-Partial and FedAvg-SSL-All. In FedAvg-SSL-Partial, the server pre-selects a subset of gateways for each training round. In FedAvg-SSL-All, all gateways participate in every round. For consistency with our default participation proportion, we configure FedAvg-SSL-Partial to select exactly 6 gateways per round.

b) **FedAsync-SSL:** FedAsync [41] is a standard asynchronous federated learning algorithm recently applied to IoT anomaly detection [15]. The server performs a global update immediately after receiving a single local model from any arbitrary gateway. We employ the polynomial staleness function with $\alpha = 0.9$, $a = 0.5$, $t-\tau \leq 16$, and $\rho = 0.005$. The original authors identified this configuration as the optimal combination [41].

c) **Local-SSL:** Local-SSL [32] aggregates the labeled server data and the unlabeled gateway data for centralized semi-supervised training. This method serves as an upper-bound reference to quantify the performance gap between our distributed framework and an idealized centralized setting.

d) **(FL)^2:** (FL)^2 [38] is a recent baseline for federated semi-supervised learning with server-side labels. This method targets configurations where only the server holds a limited amount of labeled data. Its core design includes client-specific adaptive thresholding for pseudo-label generation, sharpness-aware consistency regularization to reduce confirmation bias, and learning-status-aware aggregation to reweight local updates based on learning progress. We include (FL)^2 because its problem formulation closely matches our setting.

e) **Liu-SSL:** Liu et al. [39] propose another recent method targeting semi-supervised federated learning with server-side labels. This approach exploits clean server knowledge to guide local training and safely leverages low-confidence client samples instead of discarding them. The authors introduce a representation alignment mechanism based on server class proxies to mitigate model drift under non-IID data. They also incorporate a shrink loss to utilize unreliable pseudo-labels cautiously. This method serves as a strong comparative baseline given its highly consistent formulation.

f) **FedKD-IDS:** FedKD-IDS [30] is a recent application-oriented method for IoT federated intrusion detection. Unlike parameter-aggregation methods, this approach

TABLE VIII
PERFORMANCE COMPARISON ON CIC-IDS2017 AND EDGE-IIOTSET UNDER THE BASIC AND BALANCED SCENARIOS

Dataset	Scenario	Method	Acc.	Prec.	Recall	F1	FPR	ART	ACO
CIC-IDS2017	Basic	FedS^{3}A	**0.9818**	**0.9818**	**0.8965**	**0.9312**	**0.0059**	221.4	**0.49**
		FedAvg-SSL-Partial	0.9471	0.9471	0.7987	0.8530	0.0176	338.6	1
		FedAvg-SSL-All	0.9636	0.9636	0.8382	0.8871	0.0101	341.65	1
		FedAsync-SSL	0.8667	0.8939	0.6594	0.7510	0.0201	85.4	1
		$(FL)^2$	0.9724	0.9724	0.8639	0.9149	0.0078	356.9	1
		Liu-SSL	0.9748	0.9748	0.8724	0.9208	0.0069	349.4	1
		FedKD-IDS	0.9680	0.9680	0.8460	0.9029	0.0088	–	–
	Balanced	FedS^{3}A	**0.9819**	**0.9819**	**0.8707**	**0.9149**	**0.0029**	220.45	**0.48**
		FedAvg-SSL-Partial	0.9641	0.9641	0.8279	0.8823	0.0070	321.8	1
		FedAvg-SSL-All	0.9671	0.9671	0.8373	0.8897	0.0061	324.7	1
		FedAsync-SSL	0.9599	0.9599	0.8094	0.8707	0.0056	88.25	1
		$(FL)^2$	0.9752	0.9752	0.8605	0.9143	0.0050	340.7	1
		Liu-SSL	0.9743	0.9743	0.8568	0.9118	0.0052	334.9	1
		FedKD-IDS	0.9701	0.9701	0.8435	0.9024	0.0059	–	–
		Local-SSL	0.9910	0.9844	0.9479	0.9511	0.0016	–	–
Edge-IIoTset	Basic	FedS^{3}A	**0.9864**	**0.9864**	**0.9231**	**0.9486**	**0.0040**	84.7	**0.50**
		FedAvg-SSL-Partial	0.9668	0.9668	0.8614	0.9047	0.0105	129.8	1
		FedAvg-SSL-All	0.9728	0.9728	0.8846	0.9201	0.0082	132.1	1
		FedAsync-SSL	0.9315	0.9440	0.8091	0.8618	0.0122	33.5	1
		$(FL)^2$	0.9796	0.9796	0.8961	0.9360	0.0065	136.8	1
		Liu-SSL	0.9812	0.9812	0.9012	0.9395	0.0059	133.9	1
		FedKD-IDS	0.9748	0.9748	0.8867	0.9287	0.0076	–	–
	Balanced	FedS^{3}A	**0.9872**	**0.9872**	**0.9288**	**0.9520**	**0.0033**	83.9	**0.49**
		FedAvg-SSL-Partial	0.9731	0.9731	0.8798	0.9181	0.0080	123.6	1
		FedAvg-SSL-All	0.9760	0.9760	0.8927	0.9264	0.0068	126.8	1
		FedAsync-SSL	0.9702	0.9702	0.8719	0.9102	0.0073	34.2	1
		$(FL)^2$	0.9815	0.9815	0.9059	0.9422	0.0055	131.5	1
		Liu-SSL	0.9808	0.9808	0.9028	0.9402	0.0058	129.2	1
		FedKD-IDS	0.9775	0.9775	0.8948	0.9343	0.0064	–	–
		Local-SSL	0.9940	0.9895	0.9681	0.9728	0.0010	–	–

adopts a knowledge distillation framework where collaborators exchange logits. This method employs a voting-based aggregation strategy and incorporates an anti-poisoning verification mechanism. We include FedKD-IDS to compare FedS^{3}A against an application-specific design rather than just generic federated learning algorithms.

FedAvg-SSL and FedAsync-SSL were originally designed for traditional supervised federated learning, where the server holds no data and client datasets are fully labeled. To ensure a fair comparison under our disjoint FSSL setting, we incorporate the server's supervised update into their global optimization processes alongside the unsupervised updates from the gateways. We also apply the adaptive weight for supervised learning validated in our earlier ablation study.

Because $(FL)^2$ and Liu-SSL are inherently designed for the semi-supervised setting with server-side labels, they require minimal adaptation. We retain their core training mechanisms and evaluate them using the same data partitions, model backbones, and experimental protocols applied to FedS^{3}A.

FedKD-IDS follows a distinct collaborative paradigm relying on knowledge distillation and logit exchange. We preserve its distillation-based training logic during the evaluation to provide a robust comparison against a state-of-the-art framework explicitly designed for IoT intrusion detection.

2) Comparison Results: FedS^{3}A consistently outperforms the evaluated federated baselines on both the CIC-IDS2017 and Edge-IIoTset datasets. Compared to recent state-of-the-art semi-supervised methods such as $(FL)^2$ and Liu-SSL, FedS^{3}A achieves superior F1 scores and lower false positive rates across all configurations. Under the Basic scenario of the Edge-IIoTset dataset, the proposed framework achieves an F1 score of 0.9486. This result improves upon the performance of the strongest baseline, Liu-SSL, while maintaining a low false positive rate of 0.0040. The detection performance of FedS^{3}A closely approaches the theoretical upper bound established by the centralized Local-SSL. This proximity demonstrates the effectiveness of the proposed mechanisms for pseudo-labeling and dynamic supervised weights in distributed environments.

Experimental results also demonstrate the robustness of FedS^{3}A against non-IID data distributions. When transitioning from the Balanced scenario to the highly skewed Basic scenario, traditional asynchronous methods like FedAsync-SSL experience severe performance degradation. For example, the F1 score of FedAsync-SSL drops from 0.8707 to 0.7510 on the CIC-IDS2017 dataset. In contrast, FedS^{3}A maintains stable performance. This stability indicates that the group-based aggregation function and staleness-aware attenuation strategy effectively mitigate the model drift caused by heterogeneous client data. These mechanisms ensure reliable anomaly detection across diverse IoT networks.

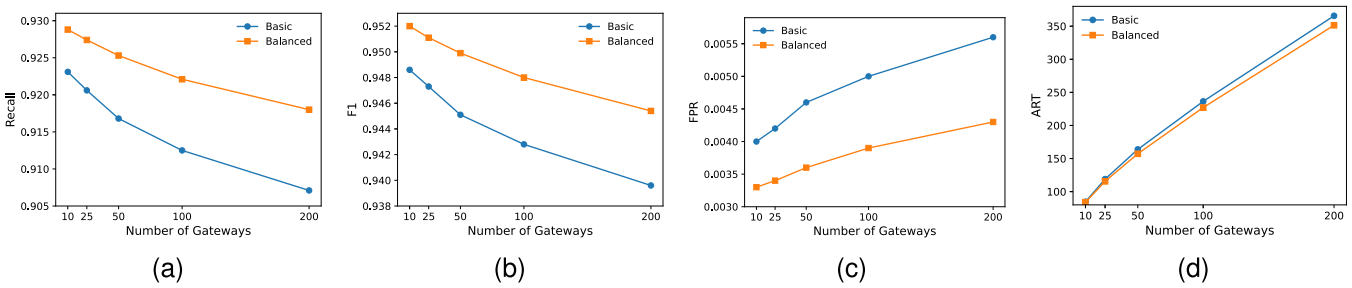


Fig. 9. The impact of the number of gateways on detection performance and round efficiency. (a) Recall. (b) F1. (c) FPR. (d) ART.

FedS^{3}A achieves a favorable balance between training efficiency and system overhead. It requires substantially less average round time than synchronous paradigms such as FedAvg-SSL-All and (FL)^2. Although FedAsync-SSL achieves the lowest average round time, it incurs a significant loss in detection accuracy. FedS^{3}A accelerates global aggregation through its semi-asynchronous mechanism without sacrificing model quality. The sparse difference transmission strategy reduces the average communication overhead to approximately 0.49. This reduction halves the bandwidth requirements compared to baselines transmitting full model parameters.

G. Scalability Analysis

1) Impact of Different Numbers of Gateways on the Edge-IIoTset Dataset: We evaluate the scalability of FedS^{3}A on the Edge-IIoTset dataset by varying the number of gateways. We repartition the training data into 10, 25, 50, 100, and 200 logical gateways and evaluate model performance under both the Basic and Balanced scenarios. This federated simulation assesses the sustained effectiveness of FedS^{3}A as the federation scales.

As shown in Fig. 9, as the number of gateways increases, the Recall and F1 scores decrease gradually across both scenarios. However, the overall degradation remains marginal. In the Basic scenario, the F1 score decreases from 0.9486 with 10 gateways to 0.9396 with 200 gateways. In the Balanced scenario, this score decreases from 0.9520 to 0.9454. The Recall score follows a similar trend. It declines from 0.9231 to 0.9071 in the Basic scenario and from 0.9288 to 0.9180 in the Balanced scenario. The Balanced scenario consistently achieves slightly higher Recall and F1 scores than the Basic scenario across all scales.

The FPR exhibits a mild increase as the number of gateways grows. The ART increases substantially in both scenarios. Under a fixed participation ratio, a larger federation requires a greater absolute number of participating gateways per round. This expansion increases the overhead of coordination, communication, and aggregation. These results indicate that FedS^{3}A degrades gracefully as the federation scales. It maintains effectiveness in larger gateway-assisted IoT configurations. However, the trade-off between detection performance and round efficiency becomes increasingly pronounced at larger scales.

VI. Conclusion

In this paper, we propose FedS^{3}A, a federated semi-supervised and semi-asynchronous learning framework for IoT anomaly detection. This framework targets a practical disjoint semi-supervised setting where only the server holds a limited amount of labeled data and the clients possess massive volumes of unlabeled data. FedS^{3}A integrates pseudo-label generation with a dynamically decaying weight for supervised learning. This combination effectively balances the contributions of server-side supervision and client-side unlabeled data throughout the training process.

To handle the inherent device heterogeneity of IoT environments, we introduce a semi-asynchronous model update mechanism paired with a staleness-tolerant distribution scheme. The server executes global aggregation upon receiving updates from a predefined fraction of clients while allowing slower clients to maintain their asynchronous training progress. FedS^{3}A scales client contributions based on local model staleness and participation frequency. The framework also employs a group-based aggregation function to mitigate the model drift caused by non-IID data distributions across clients. To optimize bandwidth, we develop a sparse difference transmission mechanism. During each communication round, clients and the server exchange only the parameter differences rather than the full model. This strategy substantially reduces communication overhead, making the framework more suitable for resource-constrained IoT and IIoT environments.

Extensive evaluations on the CIC-IDS2017 and Edge-IIoTset datasets demonstrate that FedS^{3}A consistently outperforms representative federated learning baselines in detection accuracy, round efficiency, and communication costs. Ablation studies and parameter sensitivity analyses confirm the individual contributions of the core system components. Furthermore, scalability tests on the Edge-IIoTset dataset show that the framework maintains robust detection performance as the federation size increases. These results show that FedS^{3}A provides a practical and efficient solution for anomaly detection in heterogeneous IoT networks operating under label scarcity and resource constraints.

References

[1] G. Min, L. Liu, W. Zhai, Z. Wang, and W. Lu, "An efficient data collection algorithm for partitioned wireless sensor networks," *Future Gener. Comput. Syst.*, vol. 140, pp. 53–66, Mar. 2023.

[2] M. B. M. Noor and W. H. Hassan, "Current research on Internet of Things (IoT) security: A survey," *Comput. Netw.*, vol. 148, pp. 283–294, Jan. 2019.

[3] W. Zhai, L. Liu, Y. Ding, S. Sun, and Y. Gu, "ETD: An efficient time delay attack detection framework for UAV networks," *IEEE Trans. Inf. Forensics Security*, vol. 18, pp. 2913–2928, 2023.

[4] W. Zhai, S. Sun, L. Liu, Y. Ding, and W. Lu, "HOTD: A holistic cross-layer time-delay attack detection framework for unmanned aerial vehicle networks," *J. Parallel Distrib. Comput.*, vol. 177, pp. 117–130, Jul. 2023.

[5] M. S. Pour et al., "On data-driven curation, learning, and analysis for inferring evolving Internet-of-Things (IoT) botnets in the wild," *Comput. Secur.*, vol. 91, Apr. 2020, Art. no. 101707.

[6] T. A. Ahanger, A. Aljumah, and M. Atiquzzaman, "State-of-the-art survey of artificial intelligent techniques for IoT security," *Comput. Netw.*, vol. 206, Apr. 2022, Art. no. 108771.

[7] W. Zhao, Y. Wang, W. Zhai, L. Liu, and Y. Liu, "Efficient time-delay attack detection based on node pruning and model fusion in IoT networks," *Peer-Peer Netw. Appl.*, vol. 16, no. 2, pp. 1286–1309, Mar. 2023.

[8] W. Ding, W. Zhai, L. Liu, Y. Gu, and H. Gao, "Detection of packet dropping attack based on evidence fusion in IoT networks," *Secur. Commun. Netw.*, vol. 2022, pp. 1–14, Jul. 2022.

[9] S. Hajiheidari, K. Wakil, M. Badri, and N. J. Navimipour, "Intrusion detection systems in the Internet of Things: A comprehensive investigation," *Comput. Netw.*, vol. 160, pp. 165–191, Sep. 2019.

[10] S. A. Rahman, H. Tout, C. Talhi, and A. Mourad, "Internet of Things intrusion detection: Centralized, on-device, or federated learning?" *IEEE Netw.*, vol. 34, no. 6, pp. 310–317, Nov. 2020.

[11] T. D. Nguyen, S. Marchal, M. Miettinen, H. Fereidooni, N. Asokan, and A.-R. Sadeghi, "Dïot: A federated self-learning anomaly detection system for IoT," in *Proc. IEEE 39th Int. Conf. Distrib. Comput. Syst. (ICDCS)*, Jun. 2019, pp. 756–767.

[12] V. Rey, P. M. S. Sánchez, A. H. Celdrán, and G. Bovet, "Federated learning for malware detection in IoT devices," *Comput. Netw.*, vol. 204, Feb. 2022, Art. no. 108693.

[13] E. M. Campos et al., "Evaluating federated learning for intrusion detection in Internet of Things: Review and challenges," *Comput. Netw.*, vol. 203, Feb. 2022, Art. no. 108661.

[14] Y. Liu et al., "Deep anomaly detection for time-series data in industrial IoT: A communication-efficient on-device federated learning approach," *IEEE Internet Things J.*, vol. 8, no. 8, pp. 6348–6358, Apr. 2021.

[15] P. Tian, Z. Chen, W. Yu, and W. Liao, "Towards asynchronous federated learning based threat detection: A DC-adam approach," *Comput. Secur.*, vol. 108, Sep. 2021, Art. no. 102344.

[16] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. Y. Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proc. 20th Int. Conf. Artif. Intell. Statist.*, vol. 54, A. Singh and J. Zhu. Eds., Fort Lauderdale, FL, USA, 2017, pp. 1273–1282.

[17] W. Jeong, J. Yoon, E. Yang, and S. J. Hwang, "Federated semi-supervised learning with inter-client consistency & disjoint learning," in *Proc. Int. Conf. Learn. Represent.*, 2021, pp. 1–15.

[18] E. Bagdasaryan, A. Veit, Y. Hua, D. Estrin, and V. Shmatikov, "How to backdoor federated learning," in *Proc. Int. Conf. Artif. Intell. Statist.*, 2020, pp. 2938–2948.

[19] B. Ghimire and D. B. Rawat, "Recent advances on federated learning for cybersecurity and cybersecurity for federated learning for Internet of Things," *IEEE Internet Things J.*, vol. 9, no. 11, pp. 8229–8249, Jun. 2022.

[20] S. Abdulrahman, H. Tout, A. Mourad, and C. Talhi, "FedMCCS: Multicriteria client selection model for optimal IoT federated learning," *IEEE Internet Things J.*, vol. 8, no. 6, pp. 4723–4735, Mar. 2021.

[21] W. Wu, L. He, W. Lin, R. Mao, C. Maple, and S. Jarvis, "SAFA: A semi-asynchronous protocol for fast federated learning with low overhead," *IEEE Trans. Comput.*, vol. 70, no. 5, pp. 655–668, May 2021.

[22] Q. Ma, Y. Xu, H. Xu, Z. Jiang, L. Huang, and H. Huang, "FedSA: A semi-asynchronous federated learning mechanism in heterogeneous edge computing," *IEEE J. Sel. Areas Commun.*, vol. 39, no. 12, pp. 3654–3672, Dec. 2021.

[23] Z. Zhao et al., "Federated learning with non-IID data in wireless networks," *IEEE Trans. Wireless Commun.*, vol. 21, no. 3, pp. 1927–1942, Mar. 2022.

[24] H. Wang, Z. Kaplan, D. Niu, and B. Li, "Optimizing federated learning on non-IID data with reinforcement learning," in *Proc. IEEE Conf. Comput. Commun. (IEEE INFOCOM)*, Jul. 2020, pp. 1698–1707.

[25] L. Liu, W. Zhai, X. Li, Y. Ding, W. Lu, and R. Wang, "ESTA: An efficient spatial–temporal range aggregation query processing algorithm for AAV networks," *IEEE Trans. Netw. Sci. Eng.*, vol. 13, pp. 2623–2643, 2026.

[26] J. Liu et al., "Communication-efficient asynchronous federated learning in resource-constrained edge computing," *Comput. Netw.*, vol. 199, Nov. 2021, Art. no. 108429.

[27] I. Sharafaldin, A. H. Lashkari, and A. A. Ghorbani, "Toward generating a new intrusion detection dataset and intrusion traffic characterization," *ICISSp*, vol. 1, pp. 108–116, 2018.

[28] M. A. Ferrag, O. Friha, D. Hamouda, L. Maglaras, and H. Janicke, "Edge-IIoTset: A new comprehensive realistic cyber security dataset of IoT and IIoT applications for centralized and federated learning," *IEEE Access*, vol. 10, pp. 40281–40306, 2022.

[29] Z. Jin, J. Zhou, B. Li, X. Wu, and C. Duan, "FL-IIDS: A novel federated learning-based incremental intrusion detection system," *Future Gener. Comput. Syst.*, vol. 151, pp. 57–70, Feb. 2024.

[30] N. H. Quyen, P. T. Duy, N. T. Nguyen, N. H. Khoa, and V.-H. Pham, "FedKD-IDS: A robust intrusion detection system using knowledge distillation-based semi-supervised federated learning and anti-poisoning attack mechanism," *Inf. Fusion*, vol. 117, May 2025, Art. no. 102807.

[31] R. Zhou, Z. Wang, S. Yang, D. He, and S. Chan, "Federated learning based on two-stage knowledge distillation for intrusion detection in industrial IoT," *Expert Syst. Appl.*, vol. 299, Mar. 2026, Art. no. 130144.

[32] K. Sohn et al., "FixMatch: Simplifying semi-supervised learning with consistency and confidence," in *Proc. Adv. Neural Inf. Process. Syst. (NIPS)*, 2020, pp. 596–608.

[33] W. Lu, W. Zhai, F. Wang, and Y. Fan, "A framework for moving target defense based on federated semi-supervised learning," in *Proc. Int. Conf. Electron., Comput. Commun. Technol.*, Nov. 2023, pp. 245–250.

[34] Y. Zhu, Y. Liu, J. James, and X. Yuan, "Semi-supervised federated learning for travel mode identification from GPS trajectories," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 3, pp. 2380–2391, 2022.

[35] K. Nandury, A. Mohan, and F. Weber, "Cross-silo federated training in the cloud with diversity scaling and semi-supervised learning," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, Jun. 2021, pp. 3085–3089.

[36] Z. Zhang, Y. Yang, Z. Yao, Y. Yan, J. E. Gonzalez, and M. W. Mahoney, "Improving semi-supervised federated learning by reducing the gradient diversity of models," 2020, *arXiv:2008.11364*.

[37] Y. Zhang, Z. Yang, X. Tian, N. Wang, T. Liu, and B. Han, "Robust training of federated models with extremely label deficiency," in *Proc. 12th Int. Conf. Learn. Represent.*, 2024, pp. 1–12. [Online]. Available: https://openreview.net/forum?id=qxLVaYbsSI

[38] S. Lee, T.-L. V. Le, J. Shin, and S.-J. Lee, "$(FL)^2$: Overcoming few labels in federated semi-supervised learning," in *Proc. 38th Annu. Conf. Neural Inf. Process. Syst.*, 2024, pp. 43693–43714. [Online]. Available: https://openreview.net/forum?id=lflwtGE6Vf

[39] H. Liu, Y. Mi, Y. Tang, J. Guan, and S. Zhou, "Boosting semi-supervised federated learning by effectively exploiting server-side knowledge and client-side unconfident samples," *Neural Netw.*, vol. 188, Aug. 2025, Art. no. 107440.

[40] A. Abourayya et al., "Little is enough: Boosting privacy by sharing only hard labels in federated semi-supervised learning," in *Proc. AAAI Conf. Artif. Intell.*, 2025, vol. 39, no. 15, pp. 15293–15301.

[41] C. Xie, S. Koyejo, and I. Gupta, "Asynchronous federated optimization," 2019, *arXiv:1903.03934*.

[42] Y. Chen, X. Sun, and Y. Jin, "Communication-efficient federated deep learning with layerwise asynchronous model update and temporally weighted aggregation," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 31, no. 10, pp. 4229–4238, Oct. 2020.

[43] Z. Chai, Y. Chen, A. Anwar, L. Zhao, Y. Cheng, and H. Rangwala, "FedAT: A high-performance and communication-efficient federated learning system with asynchronous tiers," in *Proc. Int. Conf. High Perform. Comput., Netw., Storage Anal.*, 2021, pp. 1–16.

[44] Y. Deng, X. Li, C. Sun, Q. Fan, X. Wang, and V. C. M. Leung, "Semi-asynchronous federated learning with trajectory prediction for vehicular edge computing," in *Proc. IEEE/ACM 32nd Int. Symp. Quality Service (IWQoS)*, Jun. 2024, pp. 1–10.

[45] C. He, S. Guo, G. Liu, and W. Zhang, "DP-SAFL: Semi-asynchronous federated learning with differential privacy in heterogeneous edge computing," *Comput. Netw.*, vol. 267, Jul. 2025, Art. no. 111346.

[46] A. Li, J. Du, D. Liu, Y. Shao, T. Zhao, and G. Ye, "CSAHFL: Clustered semi-asynchronous hierarchical federated learning for dual-layer non-IID in heterogeneous edge computing networks," in *Proc. 33rd Int. Joint Conf. Artif. Intell.*, Aug. 2024, pp. 5580–5588.

[47] Z. Zhang and M. Sabuncu, "Generalized cross entropy loss for training deep neural networks with noisy labels," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 31, 2018, pp. 8792–8802.

[48] Y. Li and Y. Liang, "Learning overparameterized neural networks via stochastic gradient descent on structured data," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 31, 2018, pp. 8168–8177.

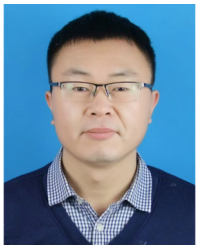
[49] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, *arXiv:1412.6980*.

[50] W. Zhai, L. Liu, J. Peng, Y. Ding, and W. Lu, "PAR: A power-aware routing algorithm for UAV networks," in *Proc. Int. Conf. Wireless Algorithms, Syst., Appl. Cham*, Switzerland: Springer, 2022, pp. 333–344.
[51] V. Costan and S. Devadas, "Intel SGX explained," *Cryptol. ePrint Archive*, 2016. [Online]. Available: https://eprint.iacr.org/2016/086
[52] M. N. Aman et al., "HAtt: Hybrid remote attestation for the Internet of Things with high availability," *IEEE Internet Things J.*, vol. 7, no. 8, pp. 7220–7233, Aug. 2020.
[53] A. N. Bhagoji, S. Chakraborty, P. P. Mittal, and S. Calp, "Analyzing federated learning through an adversarial lens," in *Proc. 36th Int. Conf. Mach. Learn.*, May 2019, pp. 634–643.
[54] E. Arazo, D. Ortego, P. Albert, N. E. O'Connor, and K. McGuinness, "Pseudo-labeling and confirmation bias in deep semi-supervised learning," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, Jul. 2020, pp. 1–8.
[55] Z. Zhou, Y. Li, X. Ren, and S. Yang, "Towards efficient and stable K-asynchronous federated learning with unbounded stale gradients on non-IID data," *IEEE Trans. Parallel Distrib. Syst.*, vol. 33, no. 12, pp. 3291–3305, Dec. 2022.
[56] S. U.Stich, "Local SGD converges fast and communicates little," in *Proc. ICLR 2019-Int. Conf. Learn. Represent.*, 2019, pp. 1–19.
[57] Z. Zhang et al., "Improving semi-supervised federated learning by reducing the gradient diversity of models," in *Proc. IEEE Int. Conf. Big Data (Big Data)*, Dec. 2021, pp. 1214–1225.
[58] G. Dudek, P. Pełka, and S. Smyl, "A hybrid residual dilated LSTM and exponential smoothing model for midterm electric load forecasting," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 33, no. 7, pp. 2879–2891, Jul. 2022.
[59] J. Konečnỳ, H. B. McMahan, F. X. Yu, P. Richtárik, A. T. Suresh, and D. Bacon, "Federated learning: Strategies for improving communication efficiency," 2016, *arXiv:1610.05492*.
[60] W. Lu, L. Liu, W. Zhai, H. Chen, and Y. Liu, "HBC: Combining lossy and lossless hybrid bilayer compression framework on time-series data," in *Proc. IEEE Int. Conf. Parallel Distrib. Process. Appl., Big Data Cloud Comput., Sustain. Comput. Commun., Social Comput. Netw. (ISPA/BDCloud/SocialCom/SustainCom)*, Dec. 2023, pp. 670–679.
[61] K. Yang et al., "DDC: Efficient dynamic-dictionary-based compression on floating time series data," in *Proc. IEEE Int. Symp. Parallel Distrib. Process. Appl. (ISPA)*, Oct. 2025, pp. 1292–1299.
[62] A. Géron, *Hands-on Machine Learning With Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques To Build Intelligent Systems*. Farnham, U.K.: O'Reilly Media, 2019.

Liang Liu received the B.S. degree in computer science from Northwestern Polytechnical University, Xi'an, Shaanxi, China, in 2005, and the Ph.D. degree in computer science from Nanjing University of Aeronautics and Astronautics, Nanjing, Jiangsu, China, in 2012. He is currently a Professor with the College of Software, Nankai University, Tianjin, China. His research interests include system software and system security.

Wenbin Zhai received the B.S. degree from Nanjing University of Chinese Medicine, Nanjing, China, in 2020, and the M.S. degree from Nanjing University of Aeronautics and Astronautics, Nanjing, in 2023. He is currently pursuing the Ph.D. degree with the Department of Computing, The Hong Kong Polytechnic University. His research interests include AI security, cybersecurity, and software security.

Feng Wang received the bachelor's degree from Zhicheng College, Fuzhou University, in 2018. He is currently pursuing the M.S. degree with the College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics. His research interests include moving target defense and distributed databases.



Youwei Ding received the B.S. and M.S. degrees in computer science from Yangzhou University, Yangzhou, Jiangsu, China, in 2007 and 2010, respectively, and the Ph.D. degree in computer science from Nanjing University of Aeronautics and Astronautics, Nanjing, Jiangsu, in 2016. He is currently a Lecturer with the School of Artificial Intelligence and Information Technology, Nanjing University of Chinese Medicine, Nanjing. His research interests include energy-efficient data management, big data analysis, and data security.

Wanying Lu received the bachelor's degree from Henan Polytechnic University in 2021. She is currently pursuing the master's degree with the College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics. Her main research interests include time series big data storage and data mining.

Weizhi Meng (Senior Member, IEEE) received the Ph.D. degree in computer science from the City University of Hong Kong, Hong Kong, in 2013. He is currently a Professor with the School of Computing and Communications, Lancaster University, Lancaster, U.K. His primary research interests include cyber security and intelligent technology in security, including intrusion detection, smartphone security, biometric authentication, HCI security, cloud security, trust management, blockchain in security, cyber–physical system security, and the IoT security. He received the Outstanding Academic Performance Award during his doctoral study, the IEEE ComSoc Best Young Researcher Award for Europe, Middle East, and Africa Region, in 2020, and the IEEE MGA Young Professionals Achievement Award in 2020. He was a recipient of Hong Kong Institution of Engineers Outstanding Paper Award for Young Engineers/Researchers in 2014 and 2017.