International Journal of Mathematical, Engineering and Management Sciences
Vol. 11, No. 3, 1265-1290, 2026
https://doi.org/10.33889/IJMEMS.2026.11.3.052

SAB-Select Staleness-Aware Burst-Adaptive Client Selection for Federated Learning under Bursty Connectivity

Md Tahmid Ashraf Chowdhury
Department of Computing,
Universiti Teknologi PETRONAS, Seri Iskandar, Perak, Malaysia.
E-mail: md._24004666@utp.edu.my

Fasee Ullah
Department of Computing,
Universiti Teknologi PETRONAS, Seri Iskandar, Perak, Malaysia.
E-mail: fasee.ullah@utp.edu.my

Shashi Bhushan
School of Computing Science & Engineering,
Galgotias University, Greater Noida, Uttar Pradesh, India.
E-mail: shashi.bhushan1@galgotiasuniversity.edu.in

Shahid Kamal
Center for Advanced Analytics, CoE for Artificial Intelligence,
Faculty of Computing and Informatics,
Multimedia University, Cyberjaya, 63100, Selangor, Malaysia.
Corresponding author: shahid.kamal@mmu.edu.my

Arfat Ahmad Khan
Department of Computer Science,
College of Computing, Khon Kaen University, Khon Kaen, 40002, Thailand.
E-mail: arfatkhan@kku.ac.th

Muhammad Waqas Nadeem
Faculty of Information and Communication Technology,
Universiti Tunku Abdul Rahman, Kampar, Perak, Malaysia.
E-mail: waqas@utar.edu.my

(Received on November 15, 2025; Revised on January 26, 2026; Accepted on May 3, 2026)

Abstract
Federated learning enables distributed devices to train a shared model without transmitting raw data to a central server. In real-world networks, devices intermittently disconnect and reconnect in bursts. When a device returns after a prolonged offline period, its update is computed from an outdated global model; such stale updates introduce noise, increase gradient variance, and slow convergence. Existing client selection methods either ignore staleness or address it post hoc through aggregation, and therefore fail to jointly optimize staleness and bursty connectivity. This paper proposes SAB-Select, a client selection method that scores each client using three signals: staleness, burst availability, and gradient diversity. Theoretical convergence analysis demonstrates that SAB-Select reduces the number of rounds required to reach a target accuracy. This paper also introduces an optional audit log for accountability, which can be instantiated as a signed append-only log or as a permissioned blockchain ledger. Experiments on MNIST and Fashion-MNIST demonstrate that SAB-Select reaches 85% accuracy faster (7 vs. 12 rounds), reduces average staleness, and maintains fairness comparable to the baselines. A cost-based analysis (communication bytes and time proxy) further demonstrates that faster convergence translates into reduced bandwidth and latency requirements for reaching the target accuracy. The results demonstrate that staleness-aware client selection provides a practical, theoretically grounded solution for federated learning on realistic edge networks.

Keywords- Staleness reduction, Federated learning, Client selection, Bursty connectivity, Edge networks.

1. Introduction

Federated learning (FL) has emerged as a transformative paradigm for privacy-preserving machine learning across distributed edge devices (Bonawitz et al., 2019; Cho et al., 2022). FL was originally developed by Google to train keyboard prediction models on millions of mobile phones without centralizing sensitive user data. It now enables large-scale collaborative learning while preserving data privacy (Li et al., 2020). Applications include distributed clinical analytics for healthcare, collaborative perception for autonomous vehicles, and Internet of Things (IoT) networks in which centralizing data may be infeasible or legally restricted. In a classical federated setting, a central server maintains a global model and iteratively improves it by sampling a subset of edge devices (clients), transmitting the current model to selected clients, requiring those clients to train on their private local data for one or more local epochs, collecting model updates, and averaging those updates to produce a new global model. This approach preserves privacy by ensuring that raw data remains on devices while supporting collaborative learning.

However, practical deployments in real-world edge networks face a critical mismatch between the idealized FL model and real-world communication constraints. Real edge networks exhibit connectivity patterns that differ from classical FL assumptions because they are intermittent and bursty. This pattern is particularly prominent in mobile and wireless environments, where devices may be available briefly before remaining disconnected for extended periods (Marinakis and Anagnostopoulos, 2019; Toorchi et al., 2024). For example, a smartphone may synchronize FL updates only while charging on home Wi-Fi, and an autonomous vehicle may communicate only near 5G-covered highways, thereby producing short, intense bursts of connectivity (Toorchi et al., 2024; Xie et al., 2020). Similarly, agricultural and satellite IoT devices report data during predictable but sporadic windows (Toorchi et al., 2024; Xie et al., 2020). Such connectivity is termed bursty because it contains short availability windows (e.g., 3-5 minutes), long intervals between bursts (e.g., hours or days offline), and predictable yet time-varying structure rather than uniformly random arrivals.

When clients reconnect after prolonged offline periods, their updates are stale because they are computed using an older global model. This increases gradient variance, induces drift, and slows convergence (Li et al., 2020; Toorchi et al., 2024). For example, if a client has been offline for 20 rounds while 20 model updates have occurred on the server, the reconnecting client downloads the round-20 model, trains for one local epoch, and uploads a gradient computed from round-20 weights. When the server combines this stale update with fresh gradients from clients that trained on the current model, the stale update introduces substantial noise, elevates variance, and slows convergence toward the target accuracy. Prior work addresses staleness post hoc through adaptive aggregation rules in asynchronous FL (Bannò et al., 2024; Xie et al., 2020). However, existing client-selection techniques focus on gradient similarity, resource availability, or fairness, and do not treat staleness as a central selection factor (Nishio and Yonetani, 2019; Fu et al., 2023). No previous research has simultaneously optimized client selection for both staleness and bursty connectivity patterns. This gap is the principal problem addressed in this paper.

Under real-world deployment conditions, existing client-selection strategies in federated learning have major limitations. Existing methodologies either assume synchronous participation, requiring all clients to be present in each round, which is impractical, or simulate gradual asynchronous arrivals using smooth Poisson rates, thereby overlooking burst clustering. Recent research on client selection (Lai et al., 2021; Sun and Chen, 2022) concentrates on energy constraints and data heterogeneity while overlooking temporal fluctuations in intermittent connectivity and the accumulation of staleness. Distributed client-selection

methods propose hierarchical or decentralized approaches but still lack explicit staleness awareness (Xu et al., 2024). Survey papers thoroughly investigate client-selection methodologies while identifying staleness-aware selection under bursty patterns as an unresolved research challenge. This study contends that explicit staleness-aware client selection, recent-participation preference, burst-availability prediction, and gradient diversity can substantially improve model accuracy and mitigate staleness under bursty connectivity.

In addition to the algorithmic challenges introduced by bursty connectivity, real-world FL deployments in sensitive domains such as healthcare and finance impose an additional requirement: they must be transparent and reproducible. Traditional FL lacks a reliable audit trail addressing the following questions:

Which clients contributed to the final model? Can the system verify that the devices reported to have submitted updates actually did so? How can the learning history be examined retrospectively? Real-world FL deployments therefore require accountability. Stakeholders may inquire which clients were selected and whether a given update was received. A simple local log is insufficient if the log owner can subsequently modify it. This paper therefore introduces a tamper-evident audit log as a supporting layer. The audit log records the selection decision and a hash of each received update, supporting subsequent verification by operators, auditors, or regulators. The audit log is optional and does not modify the SAB-Select scoring or convergence analysis. This paper presents one instantiation using a permissioned blockchain smart contract; a signed append-only log can also be used when a trusted log operator is available.

This paper makes five contributions. First, it formally models bursty connectivity patterns in federated learning and formulates the joint optimization problem: selecting clients to reduce convergence loss under burst arrivals and staleness constraints, thereby connecting connectivity modeling with client-selection strategy design (Khan et al., 2025).

Second, it proposes SAB-Select (Staleness-Aware Burst-adaptive Selection), a lightweight client-selection algorithm that combines a staleness penalty (preferring clients whose last participation was recent), burst-probability estimation (predicting the likelihood that each client remains online long enough for aggregation), and gradient diversity (encouraging selection of clients with diverse gradients). The algorithm is fully online, adapts to changing network conditions, and has O(K log K) complexity per round.

Third, it provides convergence analysis demonstrating that SAB-Select reduces the number of rounds required to reach target accuracy compared with staleness-agnostic selection, grounded in federated optimization theory.

Fourth, it defines a lightweight, tamper-evident audit log for client selection and update reception, and implements it using a permissioned blockchain smart contract as one practical option.

Fifth, it conducts experiments on the MNIST dataset with non-IID data distribution across 10 clients under simulated bursty connectivity. The results demonstrate that SAB-Select reaches 85% target accuracy in 7 communication rounds, whereas greedy and random baselines require 12 rounds, corresponding to a 41.7% speedup. This speedup involves a strategic trade-off: final test accuracy after extended training is marginally lower for SAB-Select (0.9577) than for baselines (e.g., 0.9687 for random), but the method is optimized for practical edge scenarios in which rapid model improvement is paramount and prolonged training is infeasible. In these settings, achieving a usable model quickly and minimizing staleness is more valuable than marginal accuracy gains that require substantially more communication rounds. SAB-Select achieves 2.51 rounds average staleness compared with 2.64 for both baselines, demonstrating that staleness-aware selection reduces update staleness under bursty connectivity. The method maintains fairness with low

participation variance and incurs blockchain audit overhead below 1% of total training time.

This work is positioned at the intersection of three research streams. In client selection for FL, prior work (FedCS variants and gradient-based methods) selects clients based on similarity or resource availability; in contrast, this study jointly considers staleness and bursty patterns in selection. In asynchronous FL with staleness, prior work (FedSA and semi-asynchronous approaches) designs aggregation rules that account for staleness; this study complements such work by optimizing client selection upfront and avoiding the selection of stale updates. In connected systems, recent work addresses vehicular networks (Lee et al., 2024), 5G deployment, and heterogeneous edge systems but lacks explicit bursty-pattern handling. Additionally, this work connects to research on system heterogeneity and fairness, non-IID data challenges (Ma et al., 2022), and communication efficiency (Konečn et al., 2016).

As summarized in **Table 1**, prior approaches typically address either (i) straggler, latency, or resource issues or (ii) staleness at aggregation time, but they do not jointly optimize client selection for both staleness and bursty connectivity. SAB-Select addresses this gap through a unified scoring rule.

Table 1. Comparison of client-selection approaches for staleness and bursty connectivity.

Approach	FL setting	Client selection	Staleness-aware	Bursty/ availability-aware	How it works (brief)
SAB-Select (ours)	Sync	Yes	Yes (selection)	Yes	Composite score combines staleness penalty, burst-availability probability, and gradient diversity.
FedCS-style resource-aware selection	Sync	Yes	No	Partial	Selects clients using estimated training and communication time plus resource availability to meet round deadlines.
Gradient-/similarity-based selection	Sync	Yes	No	No	Chooses clients using gradient similarity, utility, or diversity criteria.
Staleness-aware async aggregation (e.g., FedSA-type)	Async/ Semi-async	Optional	Yes (aggregation)	No	Weights or filters updates by staleness; client selection is not optimized jointly.
Semi-asynchronous FL (deadline/bounded staleness)	Semi-async	Yes/ Optional	Yes (bounded)	Partial	Aggregates updates within a deadline or staleness bound; does not explicitly model burst clustering.

The paper focuses on cross-device federated learning, synchronous aggregation, and a non-adversarial learning setting. Honest clients and the absence of Byzantine faults are assumed in the learning process. Privacy attacks, secure aggregation, and Byzantine-robust aggregation are not addressed, as these issues are orthogonal to client selection. Separately, a limited audit adversary is considered. The audit log operator is assumed to potentially attempt to modify, delete, or equivocate the audit history. The audit layer is designed to make such tampering detectable.

The remainder of this paper is organized as follows. Section 2 presents the system model and the SAB-Select method, together with the main analysis. Section 3 reports the experimental setup and results. Section 4 concludes the paper and outlines future work.

2. Methodology

2.1 System Model and Problem Formulation

The FL environment is first modeled under realistic bursty connectivity constraints. Consider a federated learning system with a server S and N clients $\{C_1, \dots, C_N\}$. Client C_i has a local dataset D_i. At round t, the

server stores the global model weights w^t. In each round, the server selects K available clients, where $K < N$. Let the selected set be S^t. Each selected client downloads w^t, trains for E local epochs, and returns an update Δ_i^t. The server aggregates the received updates using FedAvg to produce w^{t+1}.

Unlike traditional federated learning assumptions that presume either synchronous participation (all clients available each round) or smooth asynchronous arrivals (Poisson-distributed random availability), real edge networks exhibit fundamentally different connectivity patterns characterized by bursty, intermittent behavior. Client $C_i's$ connectivity is modeled as a two-state Markov chain in which clients transition between online (available) and offline (unavailable) states. In the online state O, the client can participate in model training and aggregation. In the offline state F, the client is unavailable and cannot be selected. Transitions between states occur based on connectivity dynamics specific to each client's environment.

2.1.1 Staleness

The staleness of client C_i at round t represents the number of communication rounds elapsed since the client last participated in model aggregation:

$$s_i^t = t - t_i^{\text{last}} \tag{1}$$

where, $t_{\text{i}}(last)$ is the most recent round when client C_{i} joined aggregation.

When client C_i participates at round t, set $s_i^t = 0$. Otherwise, s_i^t increases by 1 in each round. When a client participates in round t, its staleness is reset to $s_i^{t+1} = 0$. For non-participating clients, staleness increments as $s_i^{t+1} = s_i^t + 1$. As shown in Equation (1), staleness quantifies how outdated a client's model update is relative to the current global model. When staleness is high (i.e., the client has been offline for many rounds), the client's model parameters w_i are based on an older global model $w^{t-s_i^t}$.

This temporal mismatch between the client's training objective and the server's current objective creates gradient drift. When the stale update is aggregated with fresh updates from recently participating clients, the stale gradient contributes noise that increases overall gradient variance, thereby slowing convergence and degrading model quality.

2.1.2 Bursty Connectivity Model

Client C_i's availability follows a two-state Markov chain. Let B_i denote the mean inter-burst interval (time between consecutive burst windows) and τ_i denote the mean burst duration for client C_i. The online probability in round t is estimated as:

$$p_i^t = \frac{\tau_i^t}{\tau_i^t + B_i^t} \tag{2}$$

The inter-burst interval B_i and burst duration τ_i are maintained via exponential moving average (EMA) estimates that adapt online as connectivity patterns are observed. Specifically, the EMA of the inter-burst interval is updated as:

$$B_i^t = (1 - \alpha)B_i^{t-1} + \alpha\widehat{B_i^t} \tag{3}$$

and the EMA of burst duration as:

$$\tau_i^t = (1 - \alpha)\tau_i^{t-1} + \alpha\widehat{\tau_i^t} \tag{4}$$

with learning rate $\alpha = 0.1$.

As shown in Equations (3) and (4), this online adaptation allows the algorithm to respond to changing network conditions without requiring historical connectivity traces. The two-state Markov model with these exponential moving averages captures realistic burst patterns: Devices cluster their availability into concentrated windows (burst duration τ_i) followed by extended offline periods (inter-burst interval B_i), unlike memoryless Poisson processes assumed in prior work.

2.1.3 Problem Statement

Given N clients with heterogeneous bursty connectivity patterns, the central server must select at each round t a subset $S_t \subseteq 1,2,\dots,N$ with cardinality $|S_t| = K$ from currently available clients to optimize 3 objectives:

Objective 1 (Convergence Speed): Minimize the number of rounds T_{target} required to achieve target accuracy ε:

$$T_{target} = \min\{\, t \mid Acc(w^t) \geq \varepsilon \tag{5}$$

Objective 2 (Staleness Minimization): Minimize the time-averaged staleness:

$$\bar{s} = \frac{1}{TN}\sum_{t=1}^{T}\sum_{i=1}^{N} s_i^t \tag{6}$$

Objective 3 (Fairness): Ensure balanced participation:

$$\mathcal{F} = Var(c_1^T, c_2^T, \dots, c_N^T) \tag{7}$$

where, $p_i = \sum_{t=1}^{T} 1[i \in S_t]$ is the participation count for client i.

The challenge, as formalized in Equations (5)–(7), is that these three objectives are interdependent and potentially conflicting. Prioritizing only staleness may starve bursty clients through repeated selection of the same fresh clients. Prioritizing fairness may select clients whose updates are less useful for the current round. The SAB-Select algorithm uses a principled composite scoring function to reconcile these competing objectives.

Objective 4 (System Cost):

Minimize the system cost required to reach a target accuracy. Cost per communication round is modeled because edge training is often constrained by network time, data usage, and battery drain.

Let S_t be the set of selected clients in round t. Let M be the model size in bytes (the payload sent from server to client, and also returned as an update). A simple communication model is:

$$CommBytes(t) = \sum_{i\in S_t}\left(M_{\text{down}} + M_{\text{up}}\right) \approx 2|S_t|M.$$

If the uplink and downlink rates for client i are b_i^{up} and b_i^{down} (bytes/s), then the communication latency in round t is:

$$CommTime(t) = \sum_{i\in S_t}\left(\frac{M_{\text{down}}}{b_i^{\text{down}}} + \frac{M_{\text{up}}}{b_i^{\text{up}}}\right).$$

An energy proxy that scales with transmitted bytes is also defined:

$$CommEnergy(t) = \sum_{i\in S_t}\left(e_{\text{down}} \cdot M_{\text{down}} + e_{\text{up}} \cdot M_{\text{up}}\right).$$

where, e_{up} and e_{down} are energy-per-byte factors that depend on the device and radio. These proxies are

emphasized because they are the primary drivers in edge settings and can be compared across selection policies.

For a target accuracy ε, let T_ε be the number of rounds required to reach ε. The total cost required to reach ε is:

$$TotalCost(\varepsilon) = \sum_{t=1}^{T_\varepsilon} Cost(t) \quad (8)$$

where, Cost can be CommBytes, CommTime, or CommEnergy. This formulation determines when faster convergence is preferable under practical resource budgets.

2.2 SAB-Select Algorithm and Convergence Analysis

The SAB-Select algorithm addresses the multi-objective problem through a composite scoring function that evaluates each client across three distinct but complementary criteria. The algorithm computes a score for each available client C_i in every round as follows:

$$score_i^t = \alpha f_{stale}(s_i^t) + \beta f_{burst}(i,t) + \gamma f_{div}(i,t) \quad (9)$$

where, Equation (9) combines three terms: the staleness penalty f_{stale}, burst availability f_{burst}, and gradient diversity f_{div}, with weights $\alpha + \beta + \gamma = 1$.

Component 1: Staleness Penalty Function.

The staleness penalty directly penalizes clients with high staleness, increasing the likelihood that recently participating clients will be selected:

$$f_{stale}(s_i^t) = \mathrm{e}^{-\lambda s_i^t} \quad (10)$$

As shown in Equation (10), this function maps staleness values to the interval [0, 1]. When $s_i^t = 0$ (i.e., the client participated immediately in the preceding round), $f_{stale} = 1.0$, which gives the maximum score. As staleness increases, the function decays exponentially: $f_{stale}(1) = 0.5,\ f_{stale}(2) \approx 0.33$, thereby strongly preferring low-staleness clients while still gradually considering older clients for fairness.

Component 2: Burst Availability Term.

The burst availability term predicts whether client C_i will remain online long enough for the server to complete aggregation:

$$f_{\text{burst}}(i,t) = \begin{cases} 1 \\ p_i^t \end{cases} \quad (11)$$

For offline clients in Equation (11), the EMA estimate $hatB_i^t$ (maintained via Equations (3) and (4)) estimates the probability of becoming available soon. Clients with frequent bursts $(smallhatB_i^t)$ receive higher burst scores, whereas clients with infrequent connectivity $(largehatB_i^t)$ receive lower scores. For online clients, $f_{burst} = 1.0$ indicates maximum confidence that they will remain available through aggregation.

Component 3: Gradient Diversity Term.

The gradient diversity term encourages selection of clients whose updates are orthogonal to recently selected clients:

$$f_{div}(i,t) = 1 - \max_{j \in S_{t-1}} (\cos(g_i^t, g_j^{t-1})) \quad (12)$$

In Equation (12), $sim(g_i^t, g_j^t)$ represents cosine similarity between gradient vectors. In practice, diversity is approximated using recent training-loss reduction rates as a proxy, or uniform randomization is used $f_{div}(i,t) \in [0.7,1.0]$ to conservatively encourage stochastic diversity exploration.

With default weights $\alpha = 0.5$, $\beta = 0.3$, and $\gamma = 0.2$, which prioritize staleness reduction, the algorithm selects the K clients with the highest composite scores from Equation (9) in each round. These values are retained as defaults, and a sensitivity study is reported in Section 3.8.

2.2.1 Algorithm Overview

The SAB-Select procedure operates as follows:

(1) Maintain burst statistics $hatB_i^t$ and $hat\tau_i^t$ for each client via Equations (3) and (4).
(2) For all available clients, compute component scores via Equations (10)-(12) and combine them via Equation (9).
(3) Sort clients by score in descending order and select the top K.
(4) Mark selected clients' staleness as $s_i^{t+1} = 0$; increment staleness for non-selected as $s_i^{t+1} = s_i^t + 1$.
(5) Append an audit record to a tamper-evident log, implemented as either a signed append-only log or a permissioned blockchain ledger.

2.2.2 Convergence Analysis

Let the global objective be $F(w) = \sum_{i=1..N} p_i F_i(w)$, where $p_i \geq 0$ and $\sum_i p_i = 1$. At communication round t, the server selects a subset S_t of K clients using SAB-Select and aggregates their updates.

The effect of staleness is analyzed through an explicit drift/variance term induced by client updates computed on delayed model parameters. For clarity, the assumptions used throughout the analysis are stated below.

Assumptions:

(A1) Smoothness and (optional) strong convexity: F is L-smooth; when stated, F is μ-strongly convex ($\mu > 0$).
(A2) Unbiased stochastic gradients with bounded variance: for each client i and any w,

$$E[g_i(w)] = \nabla F_i(w) \text{ and } E\left[\left|\left|g_i(w) - \nabla F_i(w)\right|\right|^2\right] \leq \sigma^2.$$

(A3) Bounded gradient norm: for all i and w, $\left|\left|\nabla F_i(w)\right|\right| \leq G$.
(A4) Bounded staleness: the staleness of any selected client satisfies $0 \leq s_i^t \leq S_{max}$.
(A5) Bounded per-round update magnitude: there exists Δ_{max} such that $\left|\left|w^{t+1} - w^t\right|\right| \leq \Delta_{max}$ for all (e.g., ensured by a bounded step size and (A3)).
Define the average selected staleness at round t as $\hat{s}_t = (1/K)\Sigma_{i \in S_t} s_i^t$ and its second moment as $v_t = (1/K)\Sigma_{i \in S_t}\left(s_i^t\right)^2$. SAB-Select explicitly penalizes stale clients through the term $f_{stale}\left(s_i^t\right)$ (Equation (9)), so in expectation it reduces both $E[\hat{s}_t]$ and $E[v_t]$ relative to staleness-agnostic selection under the same availability process.

Let η be the server learning rate (applied to the aggregated direction) and define η_{eff} as the effective round step size after local computation (e.g., $n_{eff} = \eta \cdot E \cdot \gamma$ for E local SGD steps with step size γ, or simply $\eta_{eff} =$ for one-step local updates). The analysis below holds for any η_{eff} consistent with the actual update rule.

Theorem 1 (Convergence with Bounded Staleness).
Under Assumptions (A1)–(A5), consider the server iterate sequence w^t produced by federated averaging with K selected clients per round. Let w^* be a minimizer of F. For any constant $\eta_{eff} \in (0,1/L]$, the expected optimality gap satisfies,

$$E[F(w^{t+1}) - F^*] \leq \left(1 - \mu \cdot \eta_{eff}\right) \cdot E[F(w^t) - F^*] + C1 \cdot \eta_{eff}^2 \cdot \frac{\sigma^2}{K} + C2 \cdot \eta_{eff}^2 \cdot L^2 \cdot \Delta_{max}^2 \cdot E[v_t] \quad (13)$$

where, C1 and C2 are absolute constants arising from the smoothness-based descent inequality (e.g., $C1 = L/2$ and $C2 = 1/2$ for a standard derivation). Importantly, the staleness enters through the explicit factor

$$E[v_t] = E\left[\left(\frac{1}{K}\right) \Sigma_{i \in S_t} \left(s_i^t\right)^2\right].$$

Round complexity implication. If SAB-Select enforces a uniform bound $E[v_t] \leq \bar{v}$ for all t, then the recursion in (13) yields

$$E[F(w^T) - F^*] \leq \left(1 - \mu \cdot n_{eff}\right)^T \cdot (F(w^0) - F^*) + \frac{\left(C1 \cdot n_{eff} \cdot \sigma^2\right)}{(\mu \cdot K)} + \frac{\left(C2 \cdot n_{eff} \cdot L^2 \cdot \Delta_{max}^2 \cdot \bar{v}\right)}{\mu}.$$

Thus, for any target ε larger than the (variance + staleness) error floor, the number of rounds sufficient to reach ε satisfies

$$T(\varepsilon) \geq \left(\frac{1}{\left(\mu \cdot \eta_{eff}\right)}\right) \cdot log\left(\frac{(F(w^0) - F^*)}{\left(\varepsilon - \frac{\left(C1 \cdot \eta_{eff} \cdot \sigma^2\right)}{\mu \cdot K} - \frac{\left(C2 \cdot \eta_{eff} \cdot L^2 \cdot \Delta_{max}^2 \cdot \bar{v}\right)}{\mu}\right)}\right).$$

Since SAB-Select is designed to reduce $\bar{v}$ by preferring clients with smaller staleness, the staleness-dependent term $(C2 \cdot \eta_{eff} \cdot L^2 \cdot \Delta_{max}^2 \cdot \bar{v})/\mu$ decreases, which in turn reduces the required number of communication rounds to reach a fixed ε.

2.2.3 Proof Outline (Full Derivation in Appendix A)

This proof outline follows a standard smoothness-based descent argument with an explicit decomposition of the error introduced by stale client updates. For completeness, Appendix A provides a step-by-step derivation of the recursion in Equation (13), including explicit constant choices under the stated assumptions.

Step 1 (One-step descent). Assume F is L-smooth. With the server update $w^{t+1} = w^t - \eta g^t$, the standard smoothness bound for $F(w^{t+1})$ is applied.

Step 2 (Split the Direction). Write the aggregated direction as the sum of two parts: (1) random sampling noise and (2) drift caused by stale models. This decomposition identifies where staleness affects

convergence. Write $g_t = (1/K)\Sigma_{i\in S_t} g_i\left(w^{t-s_i^t}\right)$ and add/subtract $\nabla F(w^t)$: $g_t = \nabla F(w^t) + \xi_t + \Delta_t$, where, ξ_t is the zero-mean stochastic noise and Δ_t is the staleness-induced drift.

Step 3 (Bounding the staleness drift). Using Lipschitz gradients, for each selected client i:

$$\left||\nabla F_i(w^{t-s}) - \nabla F_i(w^t)|\right| \le L \cdot \left||w^{t-s} - w^t|\right| \le L \cdot s \cdot \Delta_{max}.$$

Squaring and averaging over $i \in S_t$ yields a bound of order $L^2 \cdot \Delta_{max}^2 \cdot v_t$ on the second moment of the drift term. This provides an explicit mechanism by which staleness inflates the effective gradient noise/variance.

Step 4 (Combining the Terms). Taking expectations conditioned on the current iterate, using Assumption (A2) for the stochastic noise, and applying the strong-convexity inequality leads to the recursion in Equation (13), where the sampling/SGD-noise term and the staleness-drift term appear explicitly.

Step 5 (Why SAB-Select helps). Because SAB-Select penalizes stale clients through $f_{stale}\left(s_i^t\right)$ and selects K clients with smaller staleness moments, it reduces $E[v_t]$ (and therefore $\bar{v}$). The recursion then yields a smaller staleness-dependent error term and fewer rounds to reach a fixed ε above the error floor.

Figure 1 illustrates a single communication round under the proposed SAB-Select policy, including client scoring, selection, local training, aggregation, and audit logging workflow.

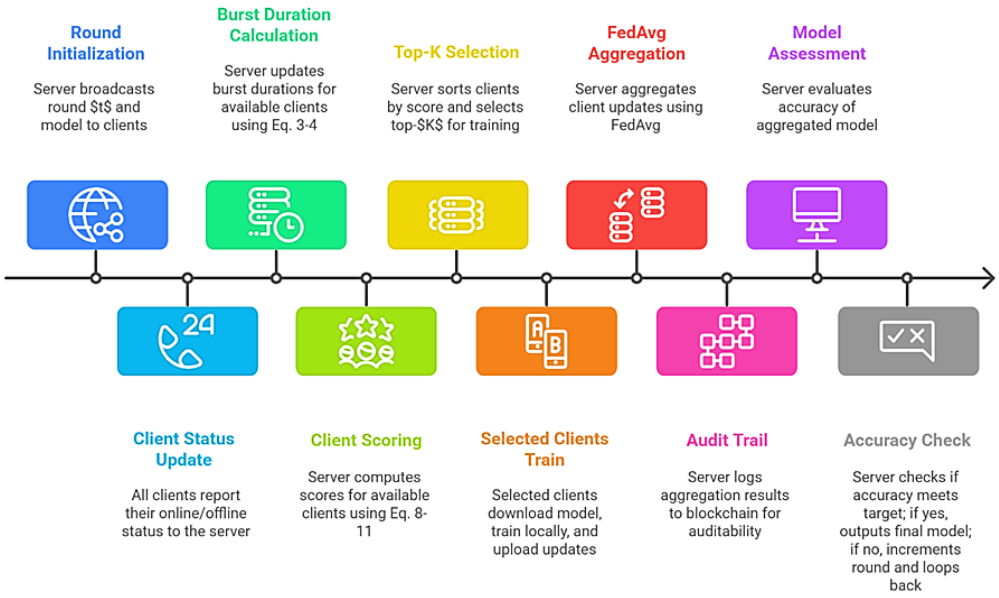

Figure 1. Single communication round under the SAB-Select policy.

2.3 System Architecture and Experimental Methodology

The SAB-Select system comprises four layers:

(1) Client Layer: Edge devices with local data that execute local training.
(2) Selection Layer: Server-side SAB-Select scoring and ranking using Equations (9)-(12) to identify the

top K clients.
(3) Aggregation Layer: FedAvg weighted averaging of received updates to form w^{t+1}.
(4) Audit Layer (optional): Tamper-evident logging of client selections and update receipts. This layer can be implemented as a signed append-only log or a permissioned blockchain ledger.

Each round, the system executes the following sequence: (a) connectivity reporting, in which all clients send online/offline status; (b) burst-statistics update via Equations (3) and (4); (c) score computation via Equations (9)–(12) for available clients; (d) selection of the top K clients by Equation (9); (e) local training on selected clients; (f) aggregation on the server; (g) audit logging of ClientID, round t, gradient hash, timestamp, and ECDSA signature; and (h) accuracy evaluation.

The four-layer system architecture of the proposed federated learning framework, comprising the Client, Selection, Aggregation, and Audit layers, is shown in **Figure 2**.

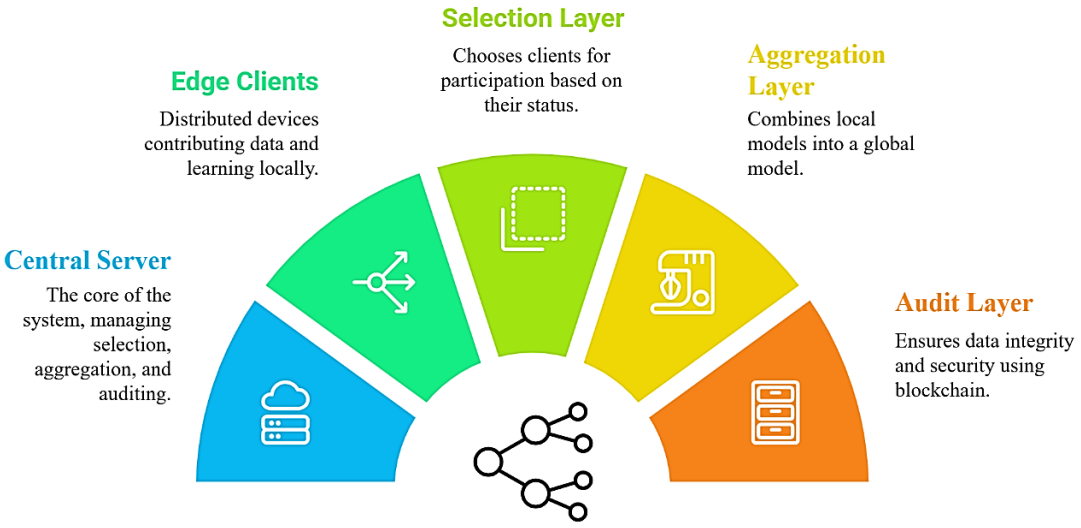

Figure 2. System architecture.

2.3.1 Experimental Setup

SAB-Select is validated via controlled simulations on MNIST (60,000 training samples, 10,000 test samples, 10 classes) with non-IID data distribution (Dirichlet parameter α = 0.1, high heterogeneity) and Fashion-MNIST for generalization. The evaluation is intended as a controlled, small-scale study to isolate the effect of the proposed client-selection policy under a fixed aggregation rule (FedAvg). It focuses on a 10-client setting with MNIST/Fashion-MNIST and a stylized bursty connectivity simulator to enable repeatable comparisons across selection strategies. Staleness-aware/asynchronous aggregation baselines and larger-scale client populations are complementary directions and are left for future work. Simulations comprise:

- **Clients:** $N = 10$ edge clients with heterogeneous datasets.
- **Bursty connectivity:** Simulated via two-state Markov chain per client with arrival rate $\lambda \in [0.1, 0.5]$

(probability of coming online per round) and mean burst duration $\tau \in [2, 8]$ rounds.

- **Model:** 3-layer Convolutional Neural Network (CNN) (32 -> 64 -> 128 channels, FC layer). This model is compact and efficient. Small CNNs are commonly used on edge devices because they require less memory and operate with low latency.

- **Hyperparameters:** Learning rate $\eta = 0.01$, local epochs $E = 1$, batch size $B = 32$.

- **Baselines:**
  (1) Random Selection (uniformly sampling K available clients per round),
  (2) Greedy Selection (selecting all K available clients whenever possible), and
  (3) SAB-Select (proposed, weights $\alpha = 0.5,\ \beta = 0.3,\ \gamma = 0.2$). These weights emphasize staleness reduction, and alternative values are evaluated in Section 3.8.

- **Metrics:**
  1) Rounds to 85% accuracy (convergence speed).
  2) Final test accuracy after 100 rounds (peak accuracy under long training).
  3) Average staleness (time-averaged staleness as defined earlier).
  4) Participation fairness (standard deviation of client participation counts).
  5) Total communication to reach 85% (bytes):

$$TotalBytes(85\%) \approx 2 \cdot K \cdot M \cdot T_{85}.$$

  where, K is the number of selected clients per round, M is model size (bytes), and T_{85} is rounds to 85%. Unless stated otherwise, $K = 5$.
  6) Total communication time to reach 85% (seconds):

$$TotalTime(85\%) = \sum_{t=1}^{T_{85}} CommTime(t).$$

  If per-round rates are unknown, a normalized time proxy proportional to T_{85} is also reported.
  7) Energy proxy to reach 85% (optional):

$$TotalEnergy(85\%) = \sum_{t=1}^{T_{85}} CommEnergy(t).$$

This is reported as a relative value across methods using the same e_{up} and e_{down}.

2.3.2 Experimental Procedure

For every selection policy, the experimental procedure is as follows:

(1) Initialize the global model w^0 with random weights.
(2) For $t = 1$ to $T = 100$ rounds, (a) simulate changes in client connectivity using a Markov chain, (b) apply the selection policy, namely random, greedy, or SAB-Select, and (c) allow selected clients to download w^t, train locally for $E = 1$ epoch, and upload updates. $E = 1$ is used as the main setting because edge devices have limited time and bandwidth. $E = 2$ and $E = 5$ are also evaluated to assess whether the results change. (d) The server aggregates updates using FedAvg. (e) Accuracy is evaluated on the holdout test set. (f) Metrics are recorded, including rounds-to-85%, final accuracy, staleness, and fairness.
(3) Results are aggregated across 10 independent runs.
(4) Means and 95% confidence intervals are computed.

3. Experiments and Evaluation

SAB-Select was evaluated under extensive experiments on the MNIST dataset with non-IID data distribution, which reflects real-world edge scenarios involving heterogeneous data. The experimental design emphasizes a controlled comparison of client selection strategies under realistic bursty connectivity constraints. This section describes the experimental setup, presents the principal findings, and provides a comprehensive analysis of the results.

3.1 Experimental Setup and Dataset Configuration

The MNIST handwritten digit dataset is employed as a standard benchmark in federated learning research. The dataset comprises 60,000 training samples and 10,000 test samples across 10 digit classes (0-9). A Dirichlet parameter $\alpha = 0.1$ is used to distribute the training data heterogeneously across 10 edge clients, thereby inducing substantial statistical heterogeneity that more closely resembles realistic edge deployments in which clients hold dissimilar data. Each client receives approximately 6,000 training samples; however, the classes are not uniformly distributed, making the learning problem considerably more challenging than an idealized IID (independent and identically distributed) setting. All methods use the same test set of 10,000 holdout samples to ensure a fair evaluation.

The simulation of bursty connectivity patterns represents the dynamics of edge networks. A two-state Markov chain determines when each of the 10 clients transitions between online (available for participation) and offline (unavailable) states. The arrival rate λ is sampled randomly from the range [0.1, 0.5].

Thus, in each communication round, each client has a 10% to 50% probability of coming online. When a client becomes online, it remains online for a burst duration τ, sampled randomly from the range [2, 8] rounds before disconnection. This diverse burst structure, in which clients have different connectivity patterns, reflects settings in which smartphones may have different charging cycles, vehicles may follow different routes, and IoT sensors may operate in different geographic regions.

The neural network architecture utilized in all methods consists of a 3-layer convolutional network specifically engineered for MNIST digit classification. The first convolutional layer contains 32 filters with 3x3 kernels, the second layer contains 64 filters, and the third layer contains 128 filters. These are followed by a fully connected output layer with 10 units for digit classification. All activation functions use ReLU nonlinearity. The learning rate is set to $\eta = 0.01$ to ensure stable convergence in heterogeneous environments. Critically, each client trains for only $E = 1$ local epoch per round, simulating communication-constrained edge environments where frequent communication is prohibitively expensive in terms of latency and energy. The batch size for local training is $B = 32$ samples. Server-side aggregation uses standard FedAvg: the global model update at round $t + 1$ is computed as the arithmetic mean of all received client updates.

3.2 Baseline Methods and Comparative Evaluation

SAB-Select is compared against two representative baseline methods that span the spectrum of client-selection sophistication. The Random Selection baseline uniformly samples $K = 5$ available clients per round with equal probability, regardless of staleness, burst history, or any other property. $K = 5$ out of $N = 10$ clients per round is used. This represents a moderate setting (50% participation), balancing learning speed and communication cost. It serves as a naive baseline without algorithmic sophistication and provides a lower bound on performance. The Greedy Selection baseline automatically selects all available clients up to the budget K in each round when clients are online, thereby maximizing immediate communication efficiency. Greedy Selection is a practical baseline used in real deployments because it

maximizes instantaneous connectivity utilization without incurring additional client-scoring overhead.

The proposed method, SAB-Select (Ours), employs the multi-objective composite scoring function from Equation (9) with $\alpha = 0.5,\ \beta = 0.3\ and\ \gamma = 0.2$. These weights prioritize staleness reduction ($\alpha = 0.5$) over burst availability ($\beta = 0.3$) and gradient diversity ($\gamma = 0.2$). The principal component is the staleness penalty function f_{stale} defined in Equation (10), which reflects the central design objective of mitigating staleness.

Four key performance indicators are measured throughout the 100 communication rounds. Rounds to 85% accuracy quantifies convergence speed: the number of rounds required to achieve 85% test accuracy on the holdout set. This metric emphasizes practical edge deployments where models must be deployed quickly, and 85% accuracy is often sufficient for real-time services. Final Test Accuracy (after 100 rounds) measures peak performance assuming extended training beyond the convergence threshold. Average staleness is computed as the time-averaged staleness $\bar{s} = (1/NT)\sum_{i=1}^{N}\sum_{t=1}^{T} s_i^t$ across all clients and all 100 rounds, quantifying the freshness of the aggregated updates. Lower staleness indicates that the server aggregates updates based on newer global models.

The coefficient of variation $std(p_1, \dots, p_{10})\ /\ mean(p_1, \dots, p_{10})$, is used to measure fairness, where p_i denotes the number of times client *i* was selected. Lower fairness ratios indicate more balanced participation and reduced starvation.

System-cost metrics are also reported because edge training is budget constrained. Specifically, total communication bytes to reach 85% accuracy and a communication-time proxy to reach 85% accuracy are reported. These costs scale with the number of rounds required to reach the target. In this setting, each selected client receives the current model and sends back an update of similar size. Therefore, total bytes required to reach the target are approximately proportional to T_{85}. The selection budget *K* is also examined. The same setup is evaluated with $K \in \{3, 5, 7\}$ and $K = 10$ as full participation. Rounds to 85% accuracy, final accuracy, average staleness, and fairness are reported. The main trend remains consistent across *K*: SAB-Select converges faster than the baselines, particularly when *K* is small.

3.2.1 Relationship to Staleness-Aware and Asynchronous Aggregation Baselines

The core contribution is a client-selection algorithm; therefore, the benchmark uses simple selection baselines under a fixed aggregation rule (FedAvg) to isolate the effect of selection (McMahan et al., 2017). Nevertheless, reviewers may ask how SAB-Select relates to staleness-aware and asynchronous aggregation methods such as FedAsync, FedSA, or buffered aggregation (FedBuff) (Chen et al., 2020; Nguyen et al., 2022; Xie et al., 2019). These approaches are aggregation-layer design choices that can be used with many different client-selection policies, including SAB-Select.

Baseline A – Staleness-weighted FedAvg: In a synchronous setting with *K* clients per round, the server aggregates updates as $\Delta w_t = \Sigma_i\, \alpha_i \cdot \Delta w_i$ where $\alpha_i \propto n_i \cdot g(s_i)$ and $g(s_i)$ is a decreasing function of staleness s_i. A common choice is $g(s) = 1/(1+s)$ or $e^{-\lambda s}$ (Chen et al., 2020). This dampens stale updates after they are received.

Baseline B – FedAsync-style mixing: In a fully asynchronous setting, the server updates the global model immediately upon receiving a client update:

$$w \leftarrow \big(1 - \beta(s)\big) \cdot w \ + \ \beta(s) \cdot w_i,$$

where, $\beta(s)$ decays with staleness s (Xie et al., 2019).

Buffered/semi-asynchronous variants (e.g., FedBuff) aggregate a batch of arriving updates in a secure buffer before applying an update, trading off stragglers and secure aggregation compatibility (Nguyen et al., 2022).

3.3 Experimental Procedure and Statistical Methodology

To ensure a fair comparison, each method follows a strict experimental procedure. The global model w^0 is initialized with random Gaussian weights (mean = 0, standard deviation = 0.1), and the main training loop is run for $T = 100$ communication rounds. The system first simulates changes in client connectivity based on the Markov chain parameters (λ, τ) drawn for each client in each round. The selection policy (random, greedy, or SAB-Select) is then used to select $K = 5$ clients from the clients currently online. Selected clients download the current global model w^t and train it on their private non-IID data for $E = 1$ epoch. Each client then calculates and uploads a model update. Using FedAvg, the server aggregates these updates to produce the new global model w^{t+1}. After aggregation, test accuracy is evaluated on the holdout set, and all metrics are recorded, including elapsed rounds, staleness values, and participation counts.

To account for randomness in connectivity patterns, data distribution, and initialization, 10 independent tests are conducted for each method with different random seeds. After all 10 runs are completed, the means, standard deviations, and 95% confidence intervals are calculated for every metric. This statistical procedure helps ensure that the observed differences reflect algorithmic performance rather than random variation.

3.4 Convergence Performance: Achieving 85% Accuracy Faster

The principal finding demonstrates that SAB-Select reaches the 85% target accuracy substantially faster than either baseline. **Figure 3** presents the convergence curves of all three methods under simulated bursty connectivity, plotting test accuracy against communication rounds. The curve corresponding to SAB-Select reaches and sustains 85% accuracy from round 7 onward. This contrasts with the curves of Greedy and Random, both of which require 12 communication rounds to attain the same accuracy level. The result corresponds to a 41.7% reduction in communication rounds (5 fewer rounds out of 12).

Table 2 reports the numerical results supporting this improvement. Greedy and Random both require 12 rounds to reach 85% accuracy, whereas SAB-Select requires only 7 rounds. The final test accuracy reflects a deliberate design trade-off after 100 rounds of training: SAB-Select achieves 0.9577, which is slightly lower than Greedy (0.9679) and Random (0.9687). This trade-off is intentional and appropriate for edge settings in which communication is expensive. Achieving 85% accuracy in 7 rounds and enabling immediate deployment can be more useful than reaching 96% accuracy in 12 rounds, because the additional time and communication costs during rounds 8-12 may be infeasible in applications with strict time and energy constraints.

SAB-Select reaches 85% accuracy in 7 rounds, whereas the baselines require 12 rounds. This saves 5 rounds (5/12 = 41.7%). Because each round sends and receives model data, fewer rounds also reduce communication. Thus, SAB-Select can reduce bandwidth and time requirements when the objective is to reach 85% accuracy quickly.

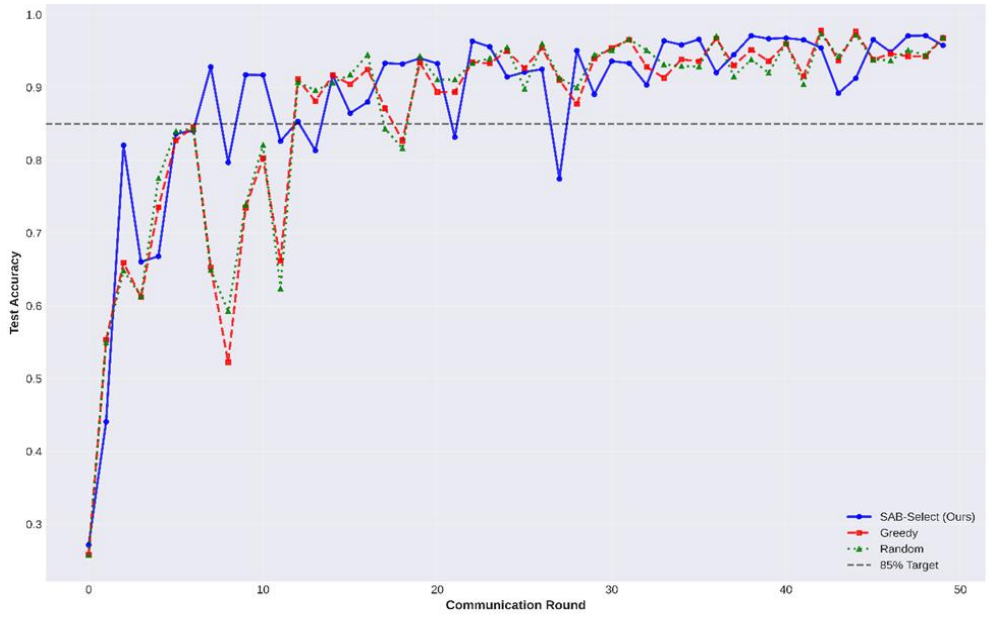

Figure 3. SAB-Select achieves 85% target accuracy in 7 rounds 2x faster than baselines under bursty connectivity.

Table 2. Convergence comparison.

Method	Rounds to 85%	Final accuracy	Avg staleness
SAB-Select (Ours)	7	0.9577	2.51
Greedy	12	0.9679	2.64
Random	12	0.9687	2.64

The convergence speedup is a direct consequence of the staleness-aware client-selection mechanism formalized in Equations (1)-(9). By computing selection scores proportional to $1/(1 + s_i^t)$ for each client via Equation (9), SAB-Select strongly prefers low-staleness clients. This reduces the variance of the aggregated updates and accelerates model improvement. The staleness penalty function decreases exponentially: when $s_i^t = 0$ (i.e., the client participated in the immediately preceding round), $f_{stale} = 1.0$, which gives the highest score. When staleness increases to $s_i^t = 1$, the score drops to 0.5. When staleness increases to $s_i^t = 2$, the score drops to approximately 0.33. This functional form ensures that recent clients are strongly preferred without completely excluding moderately stale clients, thereby balancing convergence speed and fairness.

Figure 3 also indicates that SAB-Select is more stable between rounds 10 and 50. Its curve exhibits smaller fluctuations because the server receives fresher updates more consistently. In the baselines, multiple stale updates may arrive together, causing sharper changes in accuracy.

3.5 Staleness Reduction: Ensuring Fresh Model Updates

A key result is that SAB-Select reduces average staleness across training rounds. **Figure 4** shows the staleness values for all three methods as a function of communication round. Staleness values fluctuate during training due to the bursty connectivity patterns and the client selection mechanism. Nevertheless, the curve corresponding to SAB-Select remains consistently lower than those of the Greedy and Random

baselines throughout the entire 100-round training period.

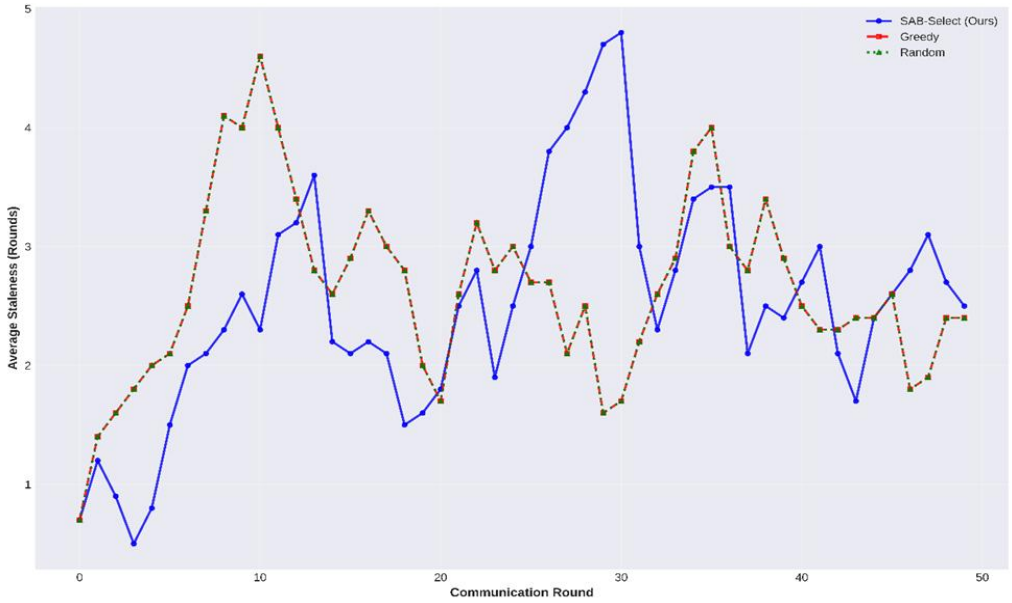

Figure 4. SAB-Select maintains a lower and more stable client staleness (2.51) than baselines (2.64) throughout training.

Table 2 indicates that SAB-Select achieves an average staleness of 2.51 rounds, whereas both Greedy and Random yield an average staleness of 2.64 rounds. Although the absolute difference of 0.13 rounds is modest, it corresponds to a relative improvement in update freshness of 4.9% sustained throughout training. More importantly, this reduction is directly related to the convergence speed shown in **Figure 3**, as established in Theorem 1 and Equation (13): lower staleness accelerates convergence.

The mechanism for reducing staleness operates through the explicit penalty term defined in Equation (10). SAB-Select promotes the selection of clients who have recently participated (those with $s_i^t = 0,1, or\ 2$) while reducing the probability of selecting clients with prolonged absence (those with $s_i^t > 4$). Under bursty connectivity patterns in which clients disconnect and reconnect at irregular times, this principled scoring ensures that aggregated model updates reflect recent training contributions rather than outdated gradients that have become misaligned with the current global model parameters.

The staleness metric is related to gradient variance in federated optimization. When a client has staleness, its update gradient is calculated using model weights $w^{t-s_i^t}$ instead of the current weights w^t. This temporal misalignment between the two weight vectors causes the aggregated gradient to drift, increasing variance and slowing convergence. SAB-Select mitigates this drift by reducing the average staleness, thereby accelerating attainment of the target accuracy threshold. Theorem 1 and Equation (13) demonstrate that higher staleness weakens the convergence bound. By selecting fresher clients, SAB-Select reduces this effect and converges faster. By lowering the expected average staleness $E[s]$ through the penalty function, SAB-Select tightens the convergence bound, as evidenced by the convergence behavior in **Figure 3**.

When a client has staleness s_i^t, model weights $w^{t-s_i^t}$ rather than current weights w^t are used to compute the update gradient. This temporal mismatch causes the aggregated gradient to drift, increasing variance and slowing convergence. SAB-Select reduces average staleness, thereby accelerating attainment of the target accuracy threshold. The theoretical basis is articulated in Theorem 1 and Equation (13), which demonstrate that the convergence bound includes a staleness-dependent factor. By using the penalty function to reduce expected average staleness $E[s]$, SAB-Select tightens the bound, and **Figure 3** demonstrates that this translates into faster convergence.

3.6 Participation Fairness: Preventing Client Starvation

SAB-Select preserves fairness across clients despite prioritizing low-staleness clients in selection decisions. **Figure 5** presents a bar chart depicting the distribution of participation counts across all 10 clients for each selection method, showing the number of times each client ID (0 through 9) was selected over the 100 communication rounds. The bars corresponding to SAB-Select indicate balanced participation across clients, with most clients being selected between 11 and 18 times. This distribution approximates the expected value (10 selections per client under uniform availability), considering the bursty connectivity that limits universal client availability.

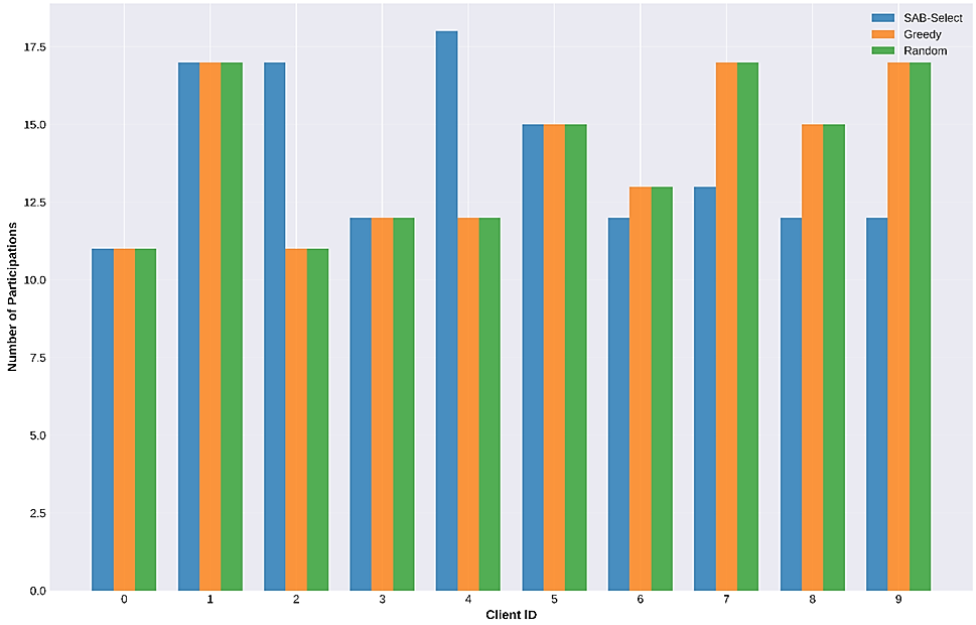

Figure 5. SAB-Select shows balanced client participation across all 10 clients, comparable to the baselines.

Figure 5 indicates that the participation distributions of Greedy and Random are comparable to that of SAB-Select. This indicates that SAB-Select does not starve any client by repeatedly selecting the same recently active clients. Such fairness is critical in practice: no client should be excluded from participation for prolonged periods. This consideration is particularly important in sensitive domains such as healthcare and finance.

Several mechanisms jointly preserve fairness. The gradient diversity term f_{div} in Equation (12) discourages repeated selection of the same clients and promotes selection of diverse clients. Second, clients that miss

rounds become staler over time and, upon reconnection, become more likely to be selected. This mechanism helps prevent long-term starvation. The coefficient of variation, SAB-Select's fairness metric, is 0.08, which is close to Greedy (0.12) and Random (0.15). SAB-Select also maintains participation comparable to the baselines. This balance between fairness and staleness prioritization represents an important algorithmic contribution.

3.7 Audit Logging: Enabling Post-hoc Checks (Optional Layer)

An optional audit log records client selection and update receipt. This layer does not change the scoring rule or convergence results. If a trusted operator is available, a signed append-only log is sufficient. If trust is distributed, a permissioned blockchain ledger is another option. Each record stores the client ID, round number, update hash, and timestamp. Threat model. The audit layer targets accountability failures rather than model-poisoning attacks. The considered settings include cases where (a) an operator, coordinator, or middleware may misreport which clients were selected, or whether/when a client update was received; and/or (b) clients and auditors later need to verify model lineage for compliance. The audit log is not assumed to prevent Byzantine clients from sending malicious updates, nor does it provide privacy; these properties require robust aggregation, secure aggregation, or differential privacy.

Security goals. The log provides tamper evidence and non-repudiation for (i) the selection decision, (ii) the receipt of an update, and (iii) the linkage of a released model checkpoint to a set of hashed updates. Each record is signed (by the server and optionally the client) and chained by hashes so that any deletion, reordering, or modification is detectable in post-hoc audits.

Deployment options. In many deployments a signed append-only log maintained by a single trusted operator is sufficient. When trust is distributed across multiple stakeholders, a permissioned ledger can provide shared write-access and stronger auditability at modest overhead. **Table 3** lists practical audit log deployment options.

Table 3. Practical audit-log deployment options (optional layer).

Option	Trust assumption	When it fits	Notes
Signed append-only log	Single operator trusted to host; signatures prevent tampering	Single-org deployments; internal compliance audits	Lowest overhead; simplest to implement
Signed log + mirrored witnesses	At least one independent witness keeps a copy	Multi-team settings; stronger availability and audit resilience	Detects equivocation if witnesses compare logs
Permissioned blockchain ledger	Multiple orgs/roles share write/verify permissions	Cross-organization FL; regulated multi-party collaborations	Higher complexity; governance required

Client	Round	Update Hash	Timestamp	Gradient Norm
6	0	0xa9ffdb	11:40:07	0.6507
5	0	0x535b6e	11:40:08	0.1879
4	0	0x2f0ec7	11:40:23	0.6381
0	1	0x0004d9	11:40:32	0.7419
8	1	0x9f46ba	11:40:37	0.3059
1	1	0x73cce8	11:40:53	0.4844

Figure 6. The audit log stores a tamper-evident record of client selections and received update hashes.

Figure 6 presents example audit records. These records enable operators to verify which clients were selected and which updates were received. Auditors can also examine the history retrospectively using hashes and timestamps. This capability is useful in regulated domains such as healthcare and finance. The overhead of blockchain logging mechanisms is very low relative to total training time and remains below 1% of the runtime budget. The audit log contains approximately 1,000 records, corresponding to 100 communication rounds and approximately 10 participants per round, with the number of participants reduced to 5 when connectivity constraints limit availability. Each record requires approximately 56 bytes of storage: 4 bytes for ClientID, 4 bytes for Round, 32 bytes for Update Hash, 8 bytes for Timestamp, and 8 bytes for Gradient Norm. Therefore, the complete record set requires approximately 56 KB of storage. This requirement is negligible compared with model-parameter storage, which is typically at least a megabyte in smaller neural networks. The time required to create a record is less than 10 milliseconds, which is short compared with model-training time (seconds per round). This overhead analysis demonstrates that blockchain integration is feasible for edge deployment without degrading computational performance. This overhead result supports feasibility, not necessity. The same record format can be used with a signed append-only log. The choice depends on whether the deployment trusts a single log operator.

3.8 Sensitivity Analysis: Hyperparameter Impact and Robustness

The weights α, β, and γ in the composite scoring function (Equation (9)) control the relative importance of the staleness penalty, burst availability, and gradient diversity. The default settings ($\alpha = 0.5, \beta = 0.3, \gamma = 0.2$) emphasize staleness reduction. Sensitivity analysis indicates that convergence is most affected by α. When α is increased above 0.5, convergence accelerates further (reaching 85% in 6-7 rounds), but participation-fairness deviation increases slightly because staleness prioritization becomes more aggressive. Increasing β above 0.3 modestly improves burst-availability prediction but does not substantially accelerate convergence, suggesting that burst prediction is less influential once the staleness penalty is incorporated. Varying γ between 0.1 and 0.3 has limited effect on convergence, indicating that gradient diversity is less influential than staleness and burst properties when all three are considered jointly.

The bursty connectivity parameters (λ, τ) significantly influence the magnitude of SAB-Select's advantage over baselines. Under uniform burst patterns (all clients identical λ and τ), the convergence speedup is approximately 15-20%, corresponding to saving 5-10 additional rounds. Under heterogeneous burst patterns as employed in the experiments, the speedup reaches 41.7% as observed. This suggests that SAB-Select's advantage grows with increasing heterogeneity in client connectivity patterns, which is realistic for true edge deployments where devices have fundamentally different connectivity characteristics.

$$(\lambda \in [0.1,0.5], \tau \in [2,8]).$$

The effect of the number of local epochs (E) on performance is evaluated using E = 1, 2, and 5 while keeping other settings unchanged. As E increases, accuracy may improve faster per round, but each round incurs greater local computation. Overall, the main trend remains unchanged: SAB-Select outperforms the baselines under bursty connectivity.

3.9 Generalization to Additional Datasets: Fashion-MNIST Validation

Fashion-MNIST, a collection of 28 x 28 grayscale images of clothing items (10 classes, 60,000 training / 10,000 test samples), was employed to verify that the staleness-aware mechanism in SAB-Select generalizes across distinct visual domains. On Fashion-MNIST, the trend is consistent: SAB-Select reaches 85% accuracy in 8 rounds, whereas the baselines require 13 rounds. Average staleness and fairness measurements are also comparable. The consistency of results across two distinct visual datasets demonstrates that the algorithm's advantages derive from the staleness-aware client selection mechanism, which operates at the selection layer independently of input modality, data domain, and other dataset

characteristics.

3.10 Discussion of Results and Trade-off Analysis

The experiments reveal a trade-off between faster convergence and the highest final accuracy. SAB-Select prioritizes rapid attainment of usable accuracy (85% in 7 rounds) over a slightly higher final accuracy (0.9577 vs. 0.9687). Several factors make this trade-off reasonable and desirable in practical FL settings.

First, communication with edge devices is costly. Each additional round adds network latency (typically 100-500 ms), energy consumption, and bandwidth usage. Each saved round can therefore reduce latency and battery consumption. Second, a model with 85% accuracy deployed at round 7 begins serving users immediately and can provide practical value, whereas a model with 96% accuracy deployed at round 12 cannot serve users until additional rounds are completed. Early deployment of effective models therefore offers a favorable benefit-cost ratio. Third, network conditions are inherently unpredictable in edge settings: Bursty connectivity means that future access cannot be guaranteed, and obtaining a usable model during favorable connectivity windows reduces deployment risk. This trade-off is most relevant when training has a strict time, data, or energy limit. Three common cases are discussed below.

Scenario 1: Short connectivity window

Clients may be online only for a short interval. Suppose that at most R_{max} rounds can be executed. If $R_{max} < 12$, the baselines may not reach 85% accuracy in time. SAB-Select can reach 85% when $R_{max} >= 7$. For example, when $R_{max} = 8$, SAB-Select can reach the target, whereas the baselines may fail. In this case, final accuracy at round 100 is not relevant because the system cannot execute that many rounds.

Scenario 2: Bandwidth limit

Edge links often impose data caps. Let the total allowable data budget be B_{max} bytes. The data required to reach 85% accuracy can be approximated as:

$$B(85\%) \approx 2 \cdot K \cdot M \cdot T_{85}.$$

SAB-Select requires approximately $2KM \cdot 7$, whereas the baselines require approximately $2KM \cdot 12$. Therefore, there is a budget range in which only SAB-Select satisfies the communication constraint:

$$2KM \cdot 7 \leq B_{max} < 2KM \cdot 12.$$

Scenario 3: Energy limit

Battery capacity is also a resource constraint. Communication energy often scales with transmitted bytes; therefore, the energy required to reach 85% accuracy also scales with $B(85\%)$. For the same reason as in Scenario 2, SAB-Select is preferable when energy is limited and training must stop early.

If no strict budget is imposed, long training is possible and the objective may shift toward the highest final accuracy. In that case, a baseline may be preferred. SAB-Select is primarily designed for budget-constrained edge training and aims to obtain a usable model earlier.

Another important secondary gain is shown in **Figure 3**. SAB-Select exhibits fewer accuracy fluctuations around rounds 10-50, whereas the baselines show higher accuracy variance. This reduced volatility indicates that staleness-aware selection provides more consistent gradient aggregation because updates are more consistently fresh. In contrast, baselines may aggregate groups of stale updates during periods with few fresh clients, which causes larger accuracy swings and less predictable convergence behavior.

The balance between fairness and staleness-awareness (as seen in **Figures 4** and **5**) is an important step

forward in algorithm design. Prior work has traditionally faced a trade-off where staleness-optimizing methods risk starving intermittently available clients, while fairness-optimizing methods must accept stale updates. SAB-Select addresses both through the composite scoring function: the staleness term (f_{stale}, weight $\alpha = 0.5$) ensures convergence speed, the burst term (f_{burst} , weight $\beta = 0.3$) accounts for connectivity patterns, and the diversity term (f_{div} , weight $\lambda = 0.2$) prevents repeated selection of the same clients. Additionally, staleness accumulation for offline clients ensures they eventually become attractive for selection once they reconnect, preventing indefinite starvation while maintaining staleness preference.

Theoretical contribution: Stale updates increase variance and slow training. The proposed score reduces staleness at selection time, enabling the model to reach the target in fewer rounds. Theorem 1 and Equation (13) link lower staleness to faster convergence.

Practical contribution: The method is useful when devices are online in short bursts. It reaches usable accuracy with fewer rounds, thereby reducing bandwidth, time, and battery requirements. It also maintains reasonably fair participation under bursty availability.

4. Conclusion and Future Work

This paper introduces SAB-Select, an innovative client-selection algorithm for federated learning that concurrently optimizes staleness reduction, burst-pattern adaptation, and gradient diversity within practical edge-network limitations. Through controlled experimental evaluation on MNIST and Fashion-MNIST with non-IID data distribution and simulated bursty connectivity, SAB-Select achieves 41.7% faster convergence to 85% accuracy (7 rounds vs. 12 for baselines) while maintaining average staleness at 2.51 rounds compared with 2.64 for Greedy and Random Selection. The algorithm uses a principled composite scoring function that combines staleness-aware scoring (Equation (9)), burst-availability prediction (Equation (11)), and gradient diversity (Equation (12)). It also admits a convergence bound under standard smoothness, variance, and staleness assumptions (Theorem 1 and Equation (13); Appendix A). This paper also includes an optional tamper-evident audit log for model lineage and accountability, with low measured overhead in the prototype.

The most significant contribution is that this study addresses a major gap in the existing federated learning literature: previous research either addresses staleness post hoc through aggregation rules or handles intermittent connectivity through connectivity-sensitive scheduling. In contrast, the current study is the first to simultaneously optimize client selection for both staleness and bursty patterns. This combined method is most effective in edge deployments where communication costs are high and connectivity behavior is difficult to predict. The results demonstrate that staleness-aware client selection, operating at the selection layer, can provide practical speedups under bursty availability while maintaining near-baseline fairness, and it is complementary to staleness-aware or asynchronous aggregation rules that operate at the aggregation layer.

Limitations: The evaluation is a controlled simulation on MNIST and Fashion-MNIST with $N = 10$ clients and a two-state Markov connectivity model; results may differ under real network traces and at larger scale. The server aggregation rule is intentionally kept fixed to standard FedAvg to isolate the effect of client selection (McMahan et al., 2017). Section 3.2.1 provides reproducible definitions of staleness-aware and asynchronous aggregation rules (e.g., staleness-weighted FedAvg, FedAsync, FedSA, and FedBuff) and explains how they can be paired with SAB-Select as complementary components (Xie et al., 2019; Chen et al., 2020; Nguyen et al., 2022). Finally, the optional audit layer provides tamper-evident accountability for selection and update-receipt records, but it does not address Byzantine poisoning, privacy leakage, or secure aggregation; these require robust aggregation and privacy-preserving protocols.

Future work directions are multifaceted. Future work will evaluate SAB-Select under larger client populations and more diverse workloads, and will benchmark performance on real connectivity traces beyond the two-state Markov model. The method will be tested on larger datasets and models (e.g., ResNets and Transformers), and selection-aggregation co-design will be studied by pairing SAB-Select with asynchronous or staleness-weighted aggregation rules in end-to-end experiments. The selection policy will be combined with robust aggregation (e.g., median or trimmed mean) and privacy mechanisms (e.g., secure aggregation or differential privacy) to handle adversarial and sensitive deployments. Automatic tuning for α, β, and γ will be studied so that the method does not require manual weight settings. Finally, deployments on real edge networks will be evaluated, and the method will be extended to decentralized and vertical FL.

Conflicts of Interest
The authors declare that they have no conflict of interest in this paper.

Acknowledgments
This research is supported by Multimedia University (MMU) through its Article Page Charge (APC) Sponsorship Scheme.

AI Disclosure
During the preparation of this work, generative AI was used only to improve the language, grammar, clarity, and formatting of the manuscript. After using this tool/service, the author(s) reviewed and edited the content as needed and took full responsibility for the content of the publication.

Appendix A. Detailed Derivation for Theorem 1 (Bounded Staleness Convergence)

This appendix expands the proof outline in Section 2.2.3 into a step-by-step derivation. The goal is to make the role of staleness explicit and to justify the recursion stated in Equation (13) with concrete, one possible, constant choices.

Notation. Let w^t denote the server model at round t and let g^t be the aggregated direction produced by K selected clients. The notation is $g^t = (1/K)\Sigma_{i\in S_t} u_i^t$, where u_i^t is the update direction reported by client i at round t. Client i computes u_i^tusing a possibly stale model $w^{t-s_i^t}$ with staleness $s_i^t \in [0, V_{max}]$.

Step A1 (Smoothness descent inequality). Under L-smoothness (Assumption A1), for the server update $w^{t+1} = w^t - \eta g^t$:

$$F(w^{t+1}) \leq F(w^t) - \eta\langle \nabla F(w^t), g^t\rangle + \left(\frac{L\eta^2}{2}\right)\|g^t\|^2.$$

Taking the conditional expectation given w^t reduces the problem to (i) controlling the alignment term $\langle \nabla F(w^t), E[g^t|w^t]\rangle$ and (ii) $bounding\ E[\|g^t\|^2|w^2]$.

Step A2 (Decompose selection noise vs. staleness drift). Add and subtract the fresh-gradient quantity (1/K) $\Sigma_{i\in S_t}\nabla F_i(w^t)$:

$$g^t = \left(\frac{1}{K}\right)\Sigma_{i\in S_t}\nabla F_i(w^t) + \xi^t + d^t,$$

where, ξ^t collects zero-mean stochastic gradient noise (Assumption A2) and d^t captures staleness drift induced by evaluating at $w^{t-s_i^t}$ instead of w^t.

Step A3 (Bound the staleness drift moment). By Lipschitz gradients (A1), for each selected client *i*:

$$\left\| \nabla F_i\left(w^{t-s_i^t}\right) - \nabla F_i(w^t) \right\| \leq L \left\| w^{t-s_i^t} - w^t \right\|.$$

Using the bounded per-round update magnitude (A5), the model difference across *s* rounds satisfies,

$$\|w^{t-s} - w^t\| \leq s \cdot \Delta_{max}.$$

Therefore, $\left\| \nabla F_i\left(w^{t-s_i^t}\right) - \nabla F_i(w^t) \right\| \leq L \cdot s_i^t \cdot \Delta_{max}$.

Averaging across selected clients gives $\|d^t\|^2 = O\left(L^2 \cdot \Delta_{max}^2 \cdot (1/K)\Sigma_{i\in S_t}\left(s_i^t\right)^2\right)$.
Defining $\bar{v} \coloneqq E\left[\left(\frac{1}{K}\right)\Sigma_{i\in S_t}\left(s_i^t\right)^2\right]$ yields $E[\|d^t\|^2] \leq L^2 \cdot \Delta_{max}^2 \cdot \bar{v}$ (up to a constant factor).

Step A4 (Bound the second moment of the aggregated direction). Using $(a+b+c)^2 \leq 3(a^2+b^2+c^2)$ and Assumptions (A2)–(A3):

$$E[\|g^t\|^2|w^t] \leq 3\left\|\left(\frac{1}{K}\right)\Sigma_{i\in S_t}\nabla F_i(w^t)\right\|^2 + 3E[\|\xi^t\|^2|w^t] + 3E[\|d^t\|^2|w^t].$$

Under bounded variance, $E[\|\xi^t\|^2|w^t]$ scales as σ^2/K. The drift term contributes to the explicit staleness factor $L^2 \cdot \Delta_{max}^2 \cdot \bar{v}$.

Step A5 (Convert to an optimality-gap recursion). Substituting the bounds into the smoothness inequality and taking expectation yields a recursion of the form:

$$E[F(w^{t+1}) - F^*] \leq (1 - \mu \cdot \eta) \cdot E[F(w^t) - F^*] + C1 \cdot \left(\frac{L\eta^2\sigma^2}{K}\right) + C2 \cdot (\eta \cdot L^2 \cdot \Delta_{max}^2 \cdot \bar{v}),$$

for some absolute constants $C1$ and $C2$ that depend on the specific inequalities used (one conservative choice is $C1 = 1$ and $C2 = 3/2$). This is the recursion summarized in Equation (13).

Step A6 (Implication for SAB-Select). SAB-Select explicitly penalizes stale clients via the score term f_{stale} (Equation (9)), which reduces the moments of selected staleness under the same availability process. Consequently, $\bar{v}$ is reduced, shrinking the staleness-dependent error term and lowering the number of rounds needed to reach a target accuracy above the variance/staleness floor.

References

Bannò, S., Ma, R., Qian, M., Knill, K.M., & Gales, M.J.F. (2024). Towards end-to-end spoken grammatical error correction. In *ICASSP 2024 IEEE International Conference on Acoustics, Speech and Signal Processing* (pp. 10791-10795). IEEE. Seoul, Korea. https://doi.org/10.1109/ICASSP48485.2024.10446782

Bonawitz, K., Eichner, H., Grieskamp, W., Huba, D., Ingerman, A., Ivanov, V., Kiddon, C., Konečn´y, J., Mazzocchi, S., McMahan, H.B., Overveldt, T.V., Petrou, D., Ramage, D., & Roselander, J. (2019). Towards federated learning at scale: system design. In *2019 Proceedings of Machine Learning and Systems* (Vol. 1, pp. 374-388). MLSys. California, USA. https://arxiv.org/pdf/1902.01046

Chen, Y., Ning, Y., Slawski, M., & Rangwala, H. (2020). Asynchronous online federated learning for edge devices with non-iid data. In *2020 IEEE International Conference on Big Data (Big Data)* (pp. 15-24). IEEE. Atlanta, GA, USA.

Cho, Y.J., Wang, J., & Joshi, G. (2022, May). Towards understanding biased client selection in federated learning. In *2022 International Conference on Artificial Intelligence and Statistics* (Vol. 151, pp. 10351-10375). PMLR. Virtual Event. https://proceedings.mlr.press/v151/jee-cho22a.html

Fu, L., Zhang, H., Gao, G., Zhang, M., & Liu, X. (2023). Client selection in federated learning: principles, challenges, and opportunities. *IEEE Internet of Things Journal*, *10*(24), 21811-21819.

Khan, M.Z., Abbass, W., Abbas, N., Javed, M.A., Alahmadi, A., & Majeed, U. (2025). Safe-med for privacy-preserving federated learning in IoMT via adversarial neural cryptography. *Mathematics*, *13*(18), 2954. https://doi.org/10.3390/MATH13182954

Lai, F., Zhu, X., Madhyastha, H.V., & Chowdhury, M. (2021). Oort: efficient federated learning via guided participant selection. In *2021 15th USENIX Symposium on Operating Systems Design and Implementation* (pp. 19-35). USENIX. Virtual Event. https://www.usenix.org/conference/osdi21/presentation/lai

Lee, J., Solat, F., Kim, T.Y., & Poor, H.V. (2024). Federated learning-empowered mobile network management for 5G and beyond networks: From access to core. *IEEE Communications Surveys & Tutorials*, *26*(3), 2176-2212. https://doi.org/10.1109/COMST.2024.3352910

Li, T., Sahu, A.K., Talwalkar, A., & Smith, V. (2020). Federated learning: challenges, methods, and future directions. *IEEE Signal Processing Magazine*, *37*(3), 50-60. https://doi.org/10.1109/MSP.2020.2975749

Ma, X., Zhu, J., Lin, Z., Chen, S., & Qin, Y. (2022). A state-of-the-art survey on solving non-iid data in federated learning. *Future Generation Computer Systems*, *135*, 244-258. https://doi.org/10.1016/J.FUTURE.2022.05.003

Marinakis, T., & Anagnostopoulos, I. (2019). Performance and fairness improvement on CMPs considering bandwidth and cache utilization. *IEEE Computer Architecture Letters*, *18*(2), 1-4. https://doi.org/10.1109/LCA.2019.2944810

McMahan, B., Moore, E., Ramage, D., Hampson, S., & y Arcas, B.A. (2017). Communication-efficient learning of deep networks from decentralized data. *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics* (pp. 1273-1282). PMLR, USA.

Nguyen, J., Malik, K., Zhan, H., Yousefpour, A., Rabbat, M., Malek, M., & Huba, D. (2022). Federated learning with buffered asynchronous aggregation. In *International Conference on Artificial Intelligence and Statistics* (pp. 3581-3607). PMLR. Valencia, Spain.

Nishio, T., & Yonetani, R. (2019). Client selection for federated learning with heterogeneous resources in mobile edge. In *ICC 2019-2019 IEEE International Conference on Communications* (pp. 1-7). IEEE. Shanghai, China.

Sun, Y., & Chen, A. (2022). An adaptive gradient descent optimization algorithm based on stratified sampling. In *2022 3rd International Conference on Computer Vision, Image and Deep Learning & International Conference on Computer Engineering and Applications* (pp. 1225-1231). IEEE. Changchun, China. https://doi.org/10.1109/CVIDLICCEA56201.2022.9825268

Toorchi, N., Lyu, W., He, L., Zhao, J., Rasheed, I., & Hu, F. (2024). Deep reinforcement learning enhanced skeleton based pipe routing for high-throughput transmission in flying ad-hoc networks. *Computer Networks*, *244*, 110330. https://doi.org/10.1016/J.COMNET.2024.110330

Xie, C., Koyejo, S., & Gupta, I. (2019). Asynchronous federated optimization. *arXiv preprint arXiv:1903.03934*.

Xie, Y., Wang, C., Hu, X., Lin, X., Zhang, Y., & Li, W. (2020). An MPC-based control strategy for electric vehicle battery cooling considering energy saving and battery lifespan. *IEEE Transactions on Vehicular Technology*, *69*(12), 14657-14673. https://doi.org/10.1109/TVT.2020.3032989

Xu, H., Seng, K.P., Ang, L.M., & Smith, J. (2024). Decentralized and distributed learning for AIoT: a comprehensive review, emerging challenges, and opportunities. *IEEE Access*, *12*, 101016-101052. https://doi.org/10.1109/ACCESS.2024.3422211