OCS-BAHFL: ONLINE COMMUNICATION SCHEDULING FOR BUFFERED ASYNCHRONOUS HIERARCHICAL FEDERATED LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Buffered asynchronous hierarchical federated learning (HFL) reduces synchronization overhead, but introduces a fundamental communication-timing problem: aggregating too frequently wastes communication and may rely on weak evidence, whereas waiting too long increases information age and staleness. We propose OCS-BAHFL, an online communication scheduling framework that independently adapts communication timing at the client–edge and edge–cloud levels. OCS-BAHFL controls an edge holding time and a cloud aggregation deadline using only event-local communication statistics, including effective evidence, event frequency, information age, and model disagreement. These observations are converted into self-normalized accumulation and freshness pressures, whose balance defines a strictly convex scalar surrogate with a closed-form waiting-time target. A projected log-domain update tracks this target while preserving positive and bounded communication timers. The scheduler requires no validation accuracy, training-loss feedback, or additional communication messages. We further establish bounded communication-induced delay, characterize controller tracking, and connect effective evidence and bounded hierarchical staleness to convergence under smooth non-convex objectives. Experiments on non-IID MNIST demonstrate that OCS-BAHFL dynamically adjusts communication timing under different local-training budgets and reduces communication volume relative to hierarchical FedAvg and a non-adaptive configuration. In particular, at the largest evaluated local-training budget, OCS-BAHFL reduces communication volume relative to hierarchical FedAvg while maintaining competitive learning performance. These results show that communication timing itself can serve as an effective online control variable in buffered asynchronous HFL.

1 INTRODUCTION

Federated learning (FL) enables distributed clients to optimize a shared model without exchanging raw data, but conventional synchronous aggregation can be limited by stragglers, communication bottlenecks, and heterogeneous computation speeds (McMahan et al., 2017). Hierarchical federated learning (HFL) alleviates part of this pressure by inserting edge aggregators between clients and the cloud, allowing updates to be consolidated locally before traversing the backbone network.

Asynchrony further improves resource utilization because clients need not complete local training simultaneously. It also creates a communication-control problem that is especially pronounced in a hierarchy. At the client–edge level, flushing a buffer too early produces frequent aggregations supported by little effective evidence, whereas waiting too long increases the age and disagreement of the buffered models. At the edge–cloud level, frequent global aggregation increases backbone communication, while excessive waiting delays already-available edge information and increases staleness. Buffered asynchronous FL (Nguyen et al., 2022) addresses part of this tradeoff in flat architectures, but hierarchical buffering creates two distinct waiting processes with different event streams and different operating conditions.

We propose **OCS-BAHFL**, an online communication scheduling framework for buffered asynchronous HFL. OCS-BAHFL adapts only two communication variables: the edge holding time

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

T_{hold}, which limits how long arrived client updates may remain in an edge buffer, and the cloud deadline $T_{\max}$, which limits how long dirty edge information may remain unconsumed before global aggregation. The two timers are updated independently from communication-side quantities already observed during asynchronous training: effective evidence, event frequency, information age, and innovation disagreement. The scheduler therefore requires neither validation/test accuracy nor training-loss feedback.

The key idea is to convert these observations into two self-normalized pressures at each hierarchy level: an *accumulation pressure*, favoring longer waiting when communication is unusually frequent or effective evidence is weak, and a *freshness pressure*, favoring shorter waiting when information is unusually old or inconsistent. A strictly convex scalar surrogate balances the two pressures and yields a closed-form timer target. Projection and log-domain tracking make the resulting updates positive, bounded, and robust to bursty asynchronous events. Staleness-aware aggregation, proximal-momentum updates, and soft hierarchical broadcast provide the learning substrate on which this scheduling mechanism operates; the principal contribution is the event-wise adaptation of the two communication timers.

Our contributions are threefold. First, we formulate buffered asynchronous HFL as two coupled but distinct communication-control processes and identify the edge holding time and cloud deadline as separate online decisions. Second, we develop a self-normalized scheduler whose closed-form updates require no additional messages, validation feedback, or manually weighted training–communication utility. Third, we establish bounded timer dynamics and show how the scheduler enters a smooth non-convex convergence bound through two complementary quantities: effective evidence controls weighted stochastic noise, while projected waiting times control hierarchical staleness. Detailed controller definitions and convergence proofs are provided in Appendices C and D, respectively.

2 Related Work

Asynchronous federated learning. Asynchronous FL removes the global synchronization barrier and allows completed client updates to be incorporated without waiting for all participants. FedAsync (Xie et al., 2019) performs immediate asynchronous aggregation, whereas FedBuff (Nguyen et al., 2022) accumulates multiple updates before a server step. These designs improve utilization but introduce stale-model error and unequal contribution rates, motivating mechanisms such as staleness-aware weighting (Liu et al., 2024). Wang et al. (Wang et al., 2024) study the joint impact of asynchronous delay and non-IID data and propose CA^{2}FL, which uses cached client updates to calibrate global aggregation. In contrast, OCS-BAHFL targets communication timing by adapting *when* buffered information is aggregated, making cached-update calibration and online communication scheduling complementary mechanisms.

Hierarchical and semi-asynchronous FL. HFL introduces intermediate edge aggregation to improve scalability and reduce cloud communication. Prior work has studied semi-asynchronous synchronization (Chen et al., 2021; 2022; Yu et al., 2023a), asynchronous hierarchical aggregation (Wang & Wang, 2022; Yu et al., 2023b), and resource-aware or learning-based coordination (Yang et al., 2023; Wu et al., 2023). More recent methods investigate adaptive aggregation, robust asynchronous coordination, clustering, and hybrid hierarchical schemes (Xia et al., 2024; Tang et al., 2025; Chen et al., 2024; He et al., 2025; Fan & Li, 2025; Li et al., 2025). These approaches primarily modify participation, aggregation policies, resource allocation, topology, robustness, or staleness handling. OCS-BAHFL instead focuses on the waiting-time decisions created by hierarchical buffering, independently adapting the client–edge holding time and edge–cloud deadline from communication-side evidence, age, and disagreement.

Position of this work. OCS-BAHFL focuses specifically on the waiting-time decisions induced by buffered asynchronous HFL. Rather than redesigning client selection or network topology, it preserves a fixed two-tier hierarchy and independently adapts communication timing at the edge and cloud levels. Both decisions rely on self-normalized statistics observed at their respective aggregation events, but govern distinct stages of the hierarchy: the edge-level decision controls how long client updates are accumulated before edge aggregation, whereas the cloud-level decision controls how long aggregated edge information waits before global aggregation. This event-specific separation distinguishes OCS-BAHFL from existing hierarchical scheduling mechanisms.

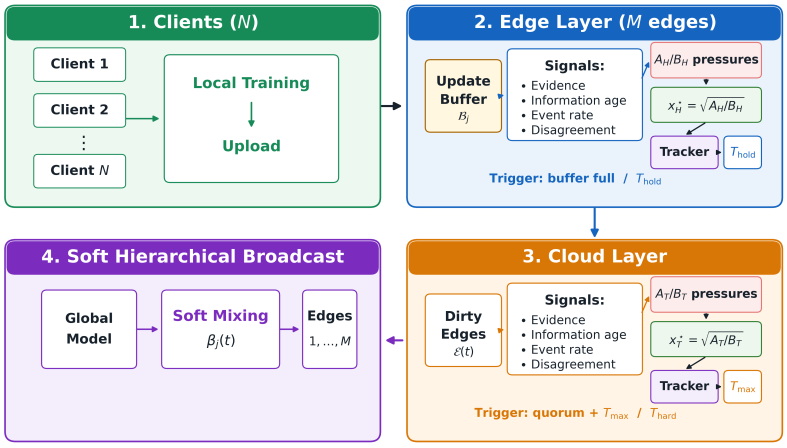


Figure 1: OCS-BAHFL architecture and online communication-control loop, showing the edge/cloud timer adaptation.

3 System Model and Asynchronous Operation

We consider a two-tier hierarchical FL system with N clients, M edge servers, and one cloud server. Client i is assigned to edge $e(i)$ and owns private dataset $\mathcal{D}_i$. The global objective is

$$F(\mathbf{w}) = \sum_{i=1}^{N} p_i F_i(\mathbf{w}), \qquad p_i = \frac{|\mathcal{D}_i|}{\sum_{n=1}^{N} |\mathcal{D}_n|}, \tag{1}$$

where F_i is the local empirical objective.

Figure 1 summarizes the OCS-BAHFL system and online communication-control loop. Clients perform local training and upload their updates to the edge layer, where updates are buffered before edge aggregation. At each scheduling level $q \in \{H, T\}$, communication signals—including effective evidence, information age, event rate, and model disagreement—form the competing A_q and B_q pressures. These pressures determine the event-local target $x_q^\star$, which is followed by the bounded online tracker to obtain the applied communication timer. At the edge, this controls the holding time T_{hold}; at the cloud, it controls the scheduled aggregation deadline $T_{\max}$. Following cloud aggregation, the global model is softly mixed with the edge models through $\beta_j(t)$. Thus, OCS-BAHFL adapts edge and cloud communication timing independently while leaving the local-training budget unchanged.

3.1 Client–Edge Operation

Clients asynchronously train from the latest model available at their associated edge and upload immediately after completion. Edge j stores arrivals in $\mathcal{B}_j(t)$. With

$$h_j(t) = t - \min_{i \in \mathcal{B}_j(t)} t_i^{\text{arr}},$$

an edge event occurs when

$$|\mathcal{B}_j(t)| \geq B_{\max} \quad \vee \quad h_j(t) \geq T_{\text{hold},j}(t). \tag{2}$$

After aggregation, the edge is marked dirty; repeated flushes overwrite its previous dirty state. We distinguish buffer residence $h_j(t)$ from end-to-end age $\tau_i(t) = t - t_i^{\text{start}}$, which is used for staleness-aware aggregation.

3.2 Edge–Cloud Operation

Let $\mathcal{E}(t)$ be the dirty-edge set and $\Delta_t^T = t - t_{\text{cloud}}^{\text{last}}$. A cloud event occurs when

$$\big(|\mathcal{E}(t)| \geq Q_{\min} \wedge \Delta_t^T \geq T_{\max}(t)\big) \vee \big(|\mathcal{E}(t)| > 0 \wedge \Delta_t^T \geq T_{\text{hard}}\big). \tag{3}$$

The first branch is the adaptive schedule, with $T_{\max}(t) = \tau_T x_T(t)$; the fixed T_{hard} is a liveness safeguard. Only the newest dirty state of each edge is retained. Thus, T_{hold} controls client-update accumulation, whereas $T_{\max}$ controls the waiting time of aggregated edge information.

The two adaptive timers therefore control different waiting stages. T_{hold} regulates the accumulation of individual client updates *before* an edge model is formed, whereas $T_{\max}$ regulates the residence of already-aggregated edge information *before* global propagation. Because the two event streams can exhibit different arrival rates, evidence levels, ages, and disagreement, OCS-BAHFL adapts these timers independently.

4 Online Communication Scheduling for Asynchronous HFL

OCS-BAHFL separates two roles. The hierarchical aggregation mechanism specifies how asynchronous information is incorporated once an event occurs, while the online scheduler determines when the next edge or cloud event should be allowed to occur.

4.1 Staleness-Aware Edge Aggregation

At an edge event, buffered client models generally have unequal data mass and unequal end-to-end age. For $i \in \mathcal{B}_j$, define

$$\alpha_i = d_i \exp(-\gamma \tau_i), \qquad p_i^H = \frac{\alpha_i}{\sum_{\ell \in \mathcal{B}_j} \alpha_\ell}, \tag{4}$$

where d_i denotes the data mass represented by client i and $\gamma \geq 0$ controls age decay. The buffered model is

$$\widetilde{\mathbf{w}}_j = \sum_{i \in \mathcal{B}_j} p_i^H \mathbf{w}_i. \tag{5}$$

A raw buffer count does not capture weight concentration. We therefore define the effective evidence

$$B_j^{\text{eff}} = \left(\sum_{i \in \mathcal{B}_j} (p_i^H)^2 \right)^{-1}. \tag{6}$$

For uniform weights, $B_j^{\text{eff}} = |\mathcal{B}_j|$; when a small number of fresh updates dominate, B_j^{eff} becomes smaller than the raw buffer size.

To reduce abrupt asynchronous drift, the edge first constructs

$$\mathbf{z}_j = \frac{\lambda_b \widetilde{\mathbf{w}}_j + \lambda_{\text{srv}} \mathbf{w}_s + \lambda_{\text{self}} \mathbf{w}_j}{\lambda_b + \lambda_{\text{srv}} + \lambda_{\text{self}}}, \tag{7}$$

and then applies

$$\mathbf{v}_j^+ = \mu_e \mathbf{v}_j + (\mathbf{z}_j - \mathbf{w}_j), \qquad \mathbf{w}_j^+ = \mathbf{w}_j + \eta_e \mathbf{v}_j^+. \tag{8}$$

The proximal target anchors the asynchronous buffered model to the current hierarchical state, while momentum smooths successive edge events. These components provide the aggregation substrate; the communication scheduler introduced below remains the principal adaptive mechanism.

4.2 Cloud Aggregation and Hierarchical Broadcast

At cloud event r, only the newest state of each participating dirty edge is used. Let m_j^r denote the evidence mass represented by edge j and $\bar{\tau}_j^r$ its weighted information age. Its cloud weight is

$$\omega_j^r = m_j^r \exp(-\gamma_s \bar{\tau}_j^r), \qquad \pi_{j,r} = \frac{\omega_j^r}{\sum_{\ell \in \mathcal{E}_r} \omega_\ell^r}. \tag{9}$$

The corresponding cloud evidence is

$$n_r^{\text{eff}} = \left(\sum_{j \in \mathcal{E}_r} \pi_{j,r}^2 \right)^{-1}. \tag{10}$$

The cloud candidate is $\widetilde{\mathbf{w}}_s^r = \sum_{j\in\mathcal{E}_r} \pi_{j,r}\mathbf{w}_j$ and is incorporated through the analogous proximal-momentum update described in Appendix B.

After a cloud event, the new global state is softly propagated:

$$\mathbf{w}_j^+ = (1-\alpha_j)\mathbf{w}_j^- + \alpha_j \mathbf{w}_s^+, \qquad 0 < \alpha_j \le 1. \tag{11}$$

Unlike hard synchronization, this retains part of the edge state and avoids forcing heterogeneous edge processes to restart from an identical model after every cloud event.

4.3 Online Communication Scheduling

The scheduling problem is an evidence–freshness tradeoff. Short waiting times increase aggregation frequency but may produce events supported by little effective evidence. Long waiting times can accumulate additional information but also increase age and model disagreement.

We normalize the physical timers as

$$x_H = \frac{T_{\text{hold}}}{\tau_H}, \qquad x_T = \frac{T_{\max}}{\tau_T}, \tag{12}$$

where τ_H and τ_T are level-specific reference time scales. The normalized timers are constrained to

$$x_H \in [\underline{x}_H, \overline{x}_H], \qquad x_T \in [\underline{x}_T, \overline{x}_T]. \tag{13}$$

At each controller event, OCS constructs two dimensionless quantities: an *accumulation pressure* A, which favors additional waiting when effective evidence is insufficient, and a *freshness pressure* B, which favors earlier aggregation when information becomes old or inconsistent.

Edge controller. At edge event k, let Δ_k^H denote the elapsed time since the previous edge event and let B_k denote the number of newly aggregated client updates. The observed update-supply rate is

$$\lambda_{H,k} = \frac{B_k}{\max(\Delta_k^H, \epsilon)}. \tag{14}$$

Using the previous supply rate $\lambda_{H,k-1}$, the code-aligned rate factor is

$$R_{H,k}^{\text{rate}} = \frac{1+\lambda_{H,k-1}\tau_H}{1+\lambda_{H,k}\tau_H}. \tag{15}$$

For desired effective evidence $n_H^\star$, weighted age $a_{H,k}$, and disagreement $D_{H,k}$, the edge pressures are

$$A_{H,k} = R_{H,k}^{\text{rate}} \frac{n_H^\star}{\max(B_k^{\text{eff}}, \epsilon)}, \qquad B_{H,k} = \frac{1+a_{H,k}/\tau_H}{1+a_{H,k-1}/\tau_H} \frac{1+D_{H,k}}{1+D_{H,k-1}}. \tag{16}$$

Hence, weak effective evidence increases $A_{H,k}$ and favors additional accumulation. In contrast, an increase in age or disagreement increases $B_{H,k}$ and favors an earlier edge event. The rate ratio adapts this decision to changes in the client-update supply process rather than relying on an absolute arrival-rate threshold.

Cloud controller. The cloud independently observes its own event stream. Let

$$\nu_{T,r} = \frac{1}{\max(\Delta_r^T, \epsilon)} \tag{17}$$

denote the cloud event frequency. Using desired cloud evidence $n_T^\star$, effective evidence n_r^{eff}, weighted age $a_{T,r}$, and disagreement $D_{T,r}$, define

$$A_{T,r} = \frac{\nu_{T,r}}{\max(\nu_{T,r-1}, \epsilon)} \frac{n_T^\star}{\max(n_r^{\text{eff}}, \epsilon)}, \qquad B_{T,r} = \frac{1+a_{T,r}/\tau_T}{1+a_{T,r-1}/\tau_T} \frac{1+D_{T,r}}{1+D_{T,r-1}}. \tag{18}$$

Frequent cloud events or insufficient effective evidence therefore increase the pressure to wait, whereas increasing age or disagreement creates pressure for earlier global propagation.

The use of ratios makes the controller self-normalized: it responds primarily to changes relative to the recent operating regime rather than to the absolute scale of communication time. Definitions of $a_{H,k}$, $a_{T,r}$, $D_{H,k}$, and $D_{T,r}$ are given in Appendix C.

Algorithm 1 OCS-BAHFL

Require: Initial models; $B_{\max}$, $Q_{\min}$, T_{hard}; timer regions $\mathcal{X}_H, \mathcal{X}_T$.
Initialize edge buffers, dirty states, T_{hold}, and $T_{\max}$.
while training is active **do**
  Clients independently train and immediately upload completed updates.
  for each edge satisfying equation 2 **do**
    Compute staleness-aware weights and B_j^{eff}.
    Perform proximal-momentum edge aggregation and mark edge dirty.
    Compute (A_H, B_H) and update T_{hold}.
  end for
  if cloud trigger equation 3 is satisfied **then**
    Aggregate dirty edges and compute n_r^{eff}.
    Update the global model and perform soft broadcast.
    Compute (A_T, B_T) and update $T_{\max}$.
    Clear the participating dirty states.
  end if
end while
return Global model $\mathbf{w}_s$.

4.4 Convex Timer Target

For either hierarchy level, OCS maps the observed pressures to a normalized waiting-time target through the communication-control surrogate

$$J(x; A, B) = \frac{A}{x} + Bx, \qquad x > 0. \tag{19}$$

The first term penalizes excessively short waiting under accumulation pressure, whereas the second penalizes excessive waiting under freshness pressure. This surrogate is not the FL training objective; rather, it provides an online control rule for balancing the two communication-side effects.

Since

$$\frac{\partial^2 J}{\partial x^2} = \frac{2A}{x^3} > 0, \tag{20}$$

J is strictly convex for $A, B > 0$. Its unique constrained minimizer is

$$x^\star = \Pi_{[\underline{x}, \overline{x}]} \sqrt{\frac{A}{B}}. \tag{21}$$

Equation equation 21 provides a direct interpretation: $A \uparrow$ increases the desired waiting time, while $B \uparrow$ decreases it. When accumulation and freshness pressures change proportionally, their ratio is preserved and the target remains unchanged.

Directly replacing x by $x^\star$ can produce oscillatory timer behavior under bursty asynchronous events. OCS instead tracks the target in the logarithmic domain:

$$x_{q,k+1} = \Pi_{\mathcal{X}_q} \left[\exp\left(\log x_{q,k} + \eta_q \operatorname{clip}\left(\log \frac{x_{q,k}^\star}{x_{q,k}}, -\Delta_q, \Delta_q \right) \right) \right], \tag{22}$$

where $q \in \{H, T\}$. Log-domain adaptation preserves positivity, projection guarantees timer feasibility, and clipping bounds the multiplicative change caused by one event.

The two recursions are asynchronous themselves: $T_{\text{hold}} = \tau_H x_H$ is updated immediately after the corresponding edge aggregation, while $T_{\max} = \tau_T x_T$ is updated after a cloud aggregation. Neither controller uses training loss, validation accuracy, or test accuracy.

OCS introduces no additional communication messages. Evidence, age, event-rate, and disagreement statistics are computed from information already available at the aggregation nodes. After these statistics are formed, timer adaptation requires only constant scalar computation and memory per controller event.

5 THEORETICAL PROPERTIES

We provide the complete theoretical analysis, assumptions, and proofs in Appendix D; here we summarize the main guarantees and their connection to the communication scheduler.

For normalized nonnegative aggregation weights $\{a_q\}_{q=1}^{K}$, define

$$N_{\text{eff}} = \left(\sum_{q=1}^{K} a_q^2 \right)^{-1}. \tag{23}$$

The effective-evidence analysis in Appendix D shows that the variance of a weighted stochastic error is bounded by σ^2/N_{eff}. Thus, effective evidence captures the stochastic averaging quality of an aggregation more accurately than its raw participant count.

Projection in the online timer update guarantees

$$\underline{x}_H \tau_H \le T_{\text{hold}} \le \overline{x}_H \tau_H, \qquad \underline{x}_T \tau_T \le T_{\max} \le \overline{x}_T \tau_T. \tag{24}$$

Hence, under bounded computation and transmission delays, the communication-induced hierarchical delay is bounded. Moreover, for a stationary target with inactive clipping and projection, the log-domain tracking error satisfies

$$\varepsilon_{q,k+1} = (1 - \eta_q)\varepsilon_{q,k}, \tag{25}$$

and therefore contracts geometrically for $0 < \eta_q \le 1$.

Finally, the convergence analysis connects communication scheduling to optimization through effective evidence and hierarchical staleness. Under the assumptions in Appendix D, after R cloud events,

$$\begin{aligned} \frac{1}{R} \sum_{r=0}^{R-1} \mathbb{E} \|\nabla F(\mathbf{w}_s^r)\|^2 \le \frac{C_0 (F(\mathbf{w}_s^0) - F_{\text{inf}})}{\eta_s R} + C_1 \eta_s \sigma^2 \frac{1}{R} \sum_{r=0}^{R-1} \mathbb{E}\left[\frac{1}{N_{\text{eff}}^r} \right] \\ + C_2 \eta_s \zeta^2 + C_3 \eta_s L^2 G^2 (\tau_{\max}^{\text{hier}})^2 + C_4 \mathcal{E}_{\text{local}} + C_5 \mathcal{E}_{\text{pm}}, \end{aligned} \tag{26}$$

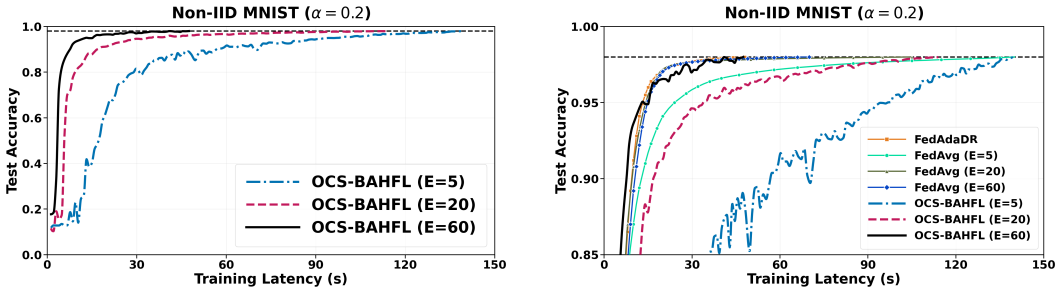
where N_{eff}^r conservatively accounts for effective evidence across both hierarchy levels. The bound makes explicit the scheduler's two main effects: effective evidence controls weighted stochastic noise, while bounded communication timers control the staleness contribution. With standard diminishing stepsizes, the optimization-dependent terms recover the usual finite-horizon rate up to the residual terms characterized in Appendix D.

6 EXPERIMENTS

6.1 EXPERIMENTAL SETUP

We follow the hierarchical FL setting of DHA-HFL (He et al., 2025) to facilitate a controlled comparison with its reported baselines. The system consists of $N = 100$ clients, $M = 10$ edge nodes, and a single cloud server, forming a three-tier client–edge–cloud architecture. Each edge node serves a disjoint subset of the clients. Following DHA-HFL, communication between neighboring edge nodes is modeled with link-success probability $p_{\text{link}} = 0.4$. We evaluate OCS-BAHFL on MNIST using the CNN architecture adopted in DHA-HFL, containing approximately 1.66 million trainable parameters. Local optimization uses SGD with learning rate 0.01 and mini-batch size 128. Statistical heterogeneity is generated using a Dirichlet partition with concentration parameter $\alpha = 0.2$, representing a strongly non-IID setting. We consider three fixed local-training budgets, $E \in \{5, 20, 60\}$, consistent with the fixed-iteration FedAvg configurations evaluated in DHA-HFL. Unless otherwise stated, the system, data, and learning settings are kept fixed across all OCS-BAHFL configurations so that the reported differences arise from the communication scheduling policy rather than changes in the underlying learning setup. OCS-BAHFL adapts the edge holding time and cloud aggregation deadline online, while E remains fixed within each experiment.

Figure 2 evaluates the convergence behavior of OCS-BAHFL on non-IID MNIST with Dirichlet parameter $\alpha = 0.2$ under three local-training budgets, $E \in \{5, 20, 60\}$. Figure 2(a) shows the



(a) OCS-BAHFL convergence

(b) Comparison near the target accuracy

Figure 2: Test accuracy versus training latency on non-IID MNIST ($\alpha = 0.2$). OCS-BAHFL is compared with fixed-E FedAvg and adaptive FedAdaDR (He et al., 2025); the latter curves are digitized from DHA-HFL. The dashed line marks the 98% target.

Table 1: Performance at 95% test accuracy on non-IID MNIST with $E = 5$. Communication volume is the cumulative model traffic required to reach the target accuracy. The minimum value in each column is shown in bold.

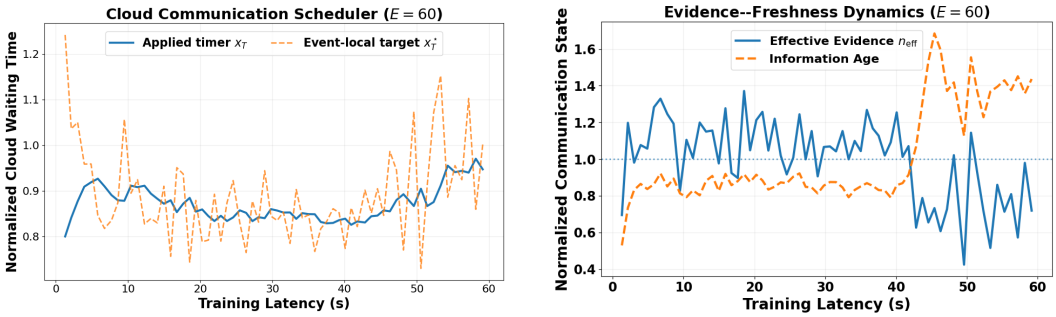
	$N = 100,\ M = 10$		$N = 50,\ M = 5$	
Method	**Latency (s)**	**Comm. (MiB)**	**Latency (s)**	**Comm. (MiB)**
OCS-BAHFL	**23.1**	22146.1	13.3	**11574.0**
Non-Adaptive	24.5	23636.2	15.1	12871.5
AChiFL	23.2	**21773.6**	13.3	11612.6
FedAvg-H	25.7	25498.9	**12.6**	12113.6

complete OCS-BAHFL trajectories from initialization to the 98% target accuracy. Increasing E substantially accelerates convergence in wall-clock training time: $E = 60$ exhibits the fastest initial convergence, followed by $E = 20$, while $E = 5$ requires considerably more time to approach the target. This behavior is expected because a larger local training budget performs more optimization work between communication events.

Figure 2(b) focuses on the high-accuracy regime and compares OCS-BAHFL with reference curves from DHA-HFL (He et al., 2025). In the DHA-HFL setting, FedAvg uses fixed local-iteration budgets $E \in \{5, 20, 60\}$, whereas FedAdaDR adaptively determines the local iteration count. The corresponding baseline curves are digitized from the published DHA-HFL results and are included as reference trajectories. Under $E = 60$, OCS-BAHFL enters the high-accuracy regime at lower latency than the fixed-$E = 60$ FedAvg reference and in a similar latency range to FedAdaDR. For $E = 20$, OCS-BAHFL reaches the target at approximately 110 s, whereas the fixed-$E = 20$ FedAvg reference reaches it substantially earlier. With $E = 5$, OCS-BAHFL and fixed-$E = 5$ FedAvg approach the target only near 140 s. Overall, the comparison indicates that the relative latency behavior depends on the local-training budget, with the strongest OCS-BAHFL behavior observed under $E = 60$.

Table 1 compares the training latency and cumulative communication required to reach 95% test accuracy with $E = 5$ under two hierarchical system scales. For $N = 100$ and $M = 10$, OCS-BAHFL reaches the target in 23.1 s, the lowest latency among the evaluated methods, while requiring 22146.1 MiB of communication. This is approximately 6.3% less communication than Non-Adaptive and 13.1% less than FedAvg-H. AChiFL achieves the lowest communication volume in this setting (21773.6 MiB), only about 1.7% below OCS-BAHFL.

For $N = 50$ and $M = 5$, OCS-BAHFL achieves the lowest communication volume, requiring 11574.0 MiB to reach the target. This represents approximately 10.1% and 4.5% lower communication than Non-Adaptive and FedAvg-H, respectively, and is slightly lower than AChiFL (11612.6 MiB). FedAvg-H reaches the target fastest in this configuration at 12.6 s, whereas OCS-BAHFL and AChiFL both require 13.3 s. Overall, OCS-BAHFL provides consistently low communication and competitive target-reaching latency across both evaluated hierarchical system scales.



(a) Cloud communication scheduler

(b) Evidence–freshness dynamics

Figure 3: Online communication-scheduling behavior for $E = 60$ on non-IID MNIST ($\alpha = 0.2$). **(a)** Applied cloud timer x_T and its event-local target $x_T^\star$ versus wall-clock time. **(b)** Normalized effective evidence and information age; the dotted line denotes the temporal mean of each signal.

Figure 3 complements the communication-volume results by examining how the online scheduler behaves during training. The trajectory plots show one representative run for visualization; aggregate communication statistics are reported separately in Table 1. Fig. 3(a) shows the cloud controller in wall-clock time. The event-local target $x_T^\star$ responds rapidly to the observed communication conditions, whereas the applied timer x_T evolves more smoothly. Importantly, $x_T^\star$ is not used directly as the waiting time. The target at event k is computed from communication statistics observed over the preceding interval and is therefore available only after that interval has completed; it can guide the subsequent timer update, but cannot causally determine the interval from which it was estimated. Moreover, directly replacing x_T by each instantaneous target would transfer event-level fluctuations into the cloud schedule. Instead, the clipped log-domain update moves x_T progressively toward $x_T^\star$, yielding the smoother trajectory observed in the figure while limiting abrupt changes in the scheduled cloud deadline. Thus, the gap between the two curves reflects the intended behavior of the online tracker rather than a failure to match the instantaneous target. This is consistent with the tracking analysis, which evaluates the online timer sequence relative to the time-varying oracle path rather than requiring pointwise equality with $x_T^\star$. Fig. 3(b) provides the corresponding evidence–freshness view. Effective evidence and information age are normalized by their respective temporal means. During the earlier part of the run, effective evidence generally remains near or above its mean while information age remains below its mean. After approximately 43 s, the two signals separate: effective evidence decreases while information age increases substantially. For the $E = 60$ run shown here, the evidence–freshness operating point changes substantially over training, illustrating the type of temporal variation that the online scheduler is designed to track.

7 Conclusion

We introduced OCS-BAHFL, an online communication scheduling framework for buffered asynchronous hierarchical federated learning that independently adapts edge holding times and cloud aggregation deadlines. Using event-local effective evidence, event frequency, information age, and model disagreement, OCS-BAHFL constructs communication-timer targets and tracks them through bounded log-domain updates without relying on validation accuracy, training loss, or additional model transmissions. The analysis establishes bounded communication waiting and online tracking properties while connecting effective evidence and hierarchical staleness to learning convergence. Experiments under non-IID data demonstrate that OCS-BAHFL consistently reduces communication relative to fixed scheduling and hierarchical FL baselines while maintaining competitive learning behavior across different local-training budgets. The observed timer and evidence–freshness dynamics further illustrate the value of adapting communication schedules to time-varying asynchronous conditions rather than relying on fixed waiting policies. Future work will evaluate the framework with larger models and datasets and under more dynamic heterogeneous environments.

AI USE STATEMENT

Large language models (LLMs) were used during the development of this work for language editing and rewriting. All AI-assisted content was reviewed and verified by the authors, who take full responsibility for the accuracy, validity, and final content of the manuscript.

REFERENCES

Qimei Chen, Zehua You, and Hao Jiang. Semi-asynchronous hierarchical federated learning for cooperative intelligent transportation systems. *arXiv preprint arXiv:2110.09073*, 2021.

Qimei Chen, Zehua You, Jing Wu, Yunpeng Liu, and Hao Jiang. Semi-asynchronous hierarchical federated learning over mobile edge networks. *IEEE Access*, 11:18887–18899, 2022.

Yonghui Chen et al. An robust secure blockchain-based hierarchical asynchronous federated learning scheme for internet of things. *IEEE Access*, 12:165280–165297, 2024.

Wei Fan and Jing Li. Joint device assignment and resource allocation for hierarchical semi-asynchronous federated learning over wireless networks. *Cluster Computing*, 28(8):522, 2025.

Yuxin He, Siteng Liao, Tong Liu, Yangguang Cui, and Shaohua Wan. Dha-hfl: A dynamic hybrid asynchronous hierarchical federated learning framework for ntn-assisted iot environments. *IEEE Internet of Things Journal*, 2025.

Aijing Li, Junping Du, Dandan Liu, Yingxia Shao, Tong Zhao, and Guanhua Ye. Csahfl: Clustered semi-asynchronous hierarchical federated learning for dual-layer non-iid in heterogeneous edge computing networks. In *Proceedings of the Thirty-Fourth International Joint Conference on Artificial Intelligence, IJCAI-25*, pp. 5580–5588. International Joint Conferences on Artificial Intelligence Organization, 2025.

Ji Liu, Juncheng Jia, Tianshi Che, Chao Huo, Jiaxiang Ren, Yang Zhou, Huaiyu Dai, and Dejing Dou. Fedasmu: Efficient asynchronous federated learning with dynamic staleness-aware model update. In *Proceedings of the AAAI conference on artificial intelligence*, volume 38, pp. 13900–13908, 2024.

H. Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Amal Agarwal. Communication-efficient learning of deep networks from decentralized data. *arXiv preprint arXiv:1602.05629*, 2017.

John Nguyen, Kshitiz Malik, Hongyuan Zhan, Ashkan Yousefpour, Mike Rabbat, Mani Malek, and Dzmitry Huba. Federated learning with buffered asynchronous aggregation. In *International conference on artificial intelligence and statistics*, pp. 3581–3607. PMLR, 2022.

Xiangyun Tang, Ruiqi Peng, Hao Li, Tao Zhang, Xiangwang Hou, and Xiangzhi Liu. Achifl: Efficient asynchronous hierarchical federated learning via adaptive edge aggregation. *IEEE Transactions on Cognitive Communications and Networking*, 12:4746–4761, 2025.

Xing Wang and Yijun Wang. Asynchronous hierarchical federated learning. *arXiv preprint arXiv:2206.00054*, 2022.

Yujia Wang, Yuanpu Cao, Jingcheng Wu, Ruoyu Chen, and Jinghui Chen. Tackling the data heterogeneity in asynchronous federated learning with cached update calibration. In *The Twelfth International Conference on Learning Representations*, 2024. URL `https://openreview.net/forum?id=4aywmeb97I`.

Qiong Wu, Xu Chen, Tao Ouyang, Zhi Zhou, Xiaoxi Zhang, Shusen Yang, and Junshan Zhang. Hiflash: Communication-efficient hierarchical federated learning with adaptive staleness control and heterogeneity-aware client-edge association. *IEEE Transactions on Parallel and Distributed Systems*, 34(5):1560–1579, 2023.

Zeke Xia, Ming Hu, Dengke Yan, Xiaofei Xie, Tianlin Li, Anran Li, Junlong Zhou, and Mingsong Chen. Cabafl: Asynchronous federated learning via hierarchical cache and feature balance. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, 43(11):4057–4068, 2024.

Cong Xie, Sanmi Koyejo, and Indranil Gupta. Asynchronous federated optimization. *arXiv preprint arXiv:1903.03934*, 2019.

Jian Yang, Yan Zhou, Wanli Wen, Jin Zhou, and Qingrui Zhang. Asynchronous hierarchical federated learning based on bandwidth allocation and client scheduling. *Applied Sciences*, 13(20): 11134, 2023.

Jieling Yu, Ruiting Zhou, Chen Chen, Bo Li, and Fang Dong. Asfl: Adaptive semi-asynchronous federated learning for balancing model accuracy and total latency in mobile edge networks. In *Proceedings of the 52nd International Conference on Parallel Processing*, pp. 443–451, 2023a.

Xiaofan Yu, Lucy Cherkasova, Harsh Vardhan, Quanling Zhao, Emily Ekaireb, Xiyuan Zhang, Arya Mazumdar, and Tajana Rosing. Async-hfl: Efficient and robust asynchronous federated learning in hierarchical iot networks. In *Proceedings of the 8th ACM/IEEE Conference on Internet of Things Design and Implementation*, pp. 236–248, 2023b.

A Complete System Semantics and Code-Aligned Event Model

This appendix expands the system details omitted from the main paper. The purpose is to make the mathematical event model correspond directly to the latest implementation rather than to an idealized synchronous-round abstraction. Time is continuous in the simulator, clients finish at different times, edge events occur independently, and cloud events are generated by a separate event process.

A.1 Client lifecycle and local optimization

Client i is permanently associated with edge $e(i)$. At launch time s_i, it snapshots the most recent model available at that edge, $\mathbf{w}_{e(i)}(s_i)$, and performs E_i local stochastic-gradient steps,

$$\mathbf{w}_i^{(q+1)} = \mathbf{w}_i^{(q)} - \eta_\ell g_i(\mathbf{w}_i^{(q)}; \xi_i^{(q)}), \qquad q = 0, \dots, E_i - 1, \tag{27}$$

with $\mathbf{w}_i^{(0)} = \mathbf{w}_{e(i)}(s_i)$. After computation and transport, the resulting model arrives at time a_i. The client is then relaunched from the edge state that is current at that new launch time. Hence, there is no client barrier and a faster client can contribute more often.

The implementation distinguishes two clocks. The end-to-end information age used in aggregation is

$$\tau_i(t) = t - s_i, \tag{28}$$

whereas the residence time used by the edge trigger begins only after arrival,

$$r_i(t) = t - a_i. \tag{29}$$

This distinction is important: local computation and network transport affect staleness weights through τ_i, but do not consume the post-arrival holding budget.

A.2 Per-edge buffers and independent holding timers

Each edge j owns a buffer $\mathcal{B}_j(t)$ and its own controller state $x_{H,j}$. Thus the physical holding time is $T_{\mathrm{hold},j} = \tau_H x_{H,j}$, not a single global edge timer. If $\mathcal{B}_j(t) \neq \varnothing$, define

$$r_j^{\max}(t) = t - \min_{i \in \mathcal{B}_j(t)} a_i. \tag{30}$$

The code-aligned edge event is

$$|\mathcal{B}_j(t)| \geq B_{\max} \quad \vee \quad r_j^{\max}(t) \geq T_{\mathrm{hold},j}(t). \tag{31}$$

An event atomically consumes the current buffer, updates the edge state, clears the buffer, stores the event time, and updates only controller j. Therefore edge controllers evolve on different event indices even though they share the same functional rule.

A.3 Dirty-state overwrite semantics

After an edge event, edge j is marked dirty and the cloud-visible record contains its newest state, represented data mass, weighted information age, and arrival time. If edge j flushes again before the next cloud event, the record is replaced rather than appended. Formally, if $\mathcal{D}(t)$ denotes the map of dirty records, an edge event applies

$$\mathcal{D}(t^+)[j] \leftarrow (\mathbf{w}_j^+, m_j, a_j^H, t), \tag{32}$$

regardless of whether key j already exists. Consequently, a high-rate edge cannot obtain multiple cloud votes merely by flushing more frequently.

A.4 Scheduled cloud route and hard-liveness route

Let $\Delta_r^T = t - t_{r-1}^T$ be the elapsed time since the previous cloud event and let $N_D(t) = |\mathcal{E}(t)|$. The implementation evaluates

$$\mathsf{Scheduled}(t) = \mathbb{I}\{N_D(t) \geq Q_{\min}\}\mathbb{I}\{\Delta_r^T \geq T_{\max}(t)\}, \tag{33}$$

$$\mathsf{Liveness}(t) = \mathbb{I}\{N_D(t) > 0\}\mathbb{I}\{\Delta_r^T \geq T_{\mathrm{hard}}\}. \tag{34}$$

A cloud event occurs when either indicator is one. Only $T_{\max}$ is an online decision. T_{hard} is a fixed safety bound, so the liveness route should not be interpreted as a second learned or optimized scheduler. When both routes are true, the event is recorded as scheduled because the normal condition already suffices.

A.5 Event ordering and state transitions

At a simulation timestamp, completed client jobs are first inserted into their associated buffers. Eligible edges are then flushed. The resulting dirty records are immediately visible to the cloud trigger evaluated at that same time. This ordering permits a newly flushed edge to contribute to a cloud event without an artificial extra time step. After a cloud event, the global model is updated, soft broadcast is applied to the participating edge states, the cloud controller is updated, the dirty map is cleared, and the next cloud inter-event clock starts.

A.6 Communication accounting

Let S_{mdl} be the serialized model size. The implementation counts executed model transmissions rather than an analytical proxy. If $N_{CE}(t)$, $N_{EC}(t)$, and $N_{BC}(t)$ are the cumulative client–edge, edge–cloud, and cloud-broadcast model transmissions, respectively, then

$$C_{\mathrm{bytes}}(t) = S_{\mathrm{mdl}}\big(N_{CE}(t) + N_{EC}(t) + N_{BC}(t)\big). \tag{35}$$

Controller updates add no model transmission; they use statistics already available at the aggregation nodes.

B Detailed Hierarchical Aggregation

This appendix collects the aggregation definitions omitted from the main text.

B.1 Client age and edge buffering

For a client that starts local training at t_i^{start} and arrives at its edge at t_i^{arr}, the end-to-end age at processing time t is

$$\tau_i(t) = t - t_i^{\mathrm{start}}. \tag{36}$$

For edge j, the residence time of the oldest buffered update is

$$a_j^{\mathrm{buf}}(t) = \begin{cases} t - \min_{i\in\mathcal{B}_j(t)} t_i^{\mathrm{arr}}, & |\mathcal{B}_j(t)| > 0, \\ 0, & |\mathcal{B}_j(t)| = 0. \end{cases} \tag{37}$$

The edge flush condition is

$$|\mathcal{B}_j(t)| \geq B_{\max} \quad \text{or} \quad a_j^{\mathrm{buf}}(t) \geq T_{\mathrm{hold}}(t). \tag{38}$$

B.2 Edge aggregation and momentum

With staleness-aware client weights $\alpha_i = d_i e^{-\gamma\tau_i}$ and $p_i^H = \alpha_i / \sum_{\ell\in\mathcal{B}_j}\alpha_\ell$, the buffered edge model is

$$\widetilde{\mathbf{w}}_j = \sum_{i\in\mathcal{B}_j} p_i^H \mathbf{w}_i, \qquad B^{\mathrm{eff}} = \frac{1}{\sum_i (p_i^H)^2}. \tag{39}$$

The proximal target is given by equation 7. The edge then applies

$$\mathbf{v}_j^+ = \mu_e \mathbf{v}_j + (\mathbf{z}_j - \mathbf{w}_j), \qquad \mathbf{w}_j^+ = \mathbf{w}_j + \eta_e \mathbf{v}_j^+. \tag{40}$$

B.3 Cloud aggregation and soft broadcast

At cloud event r, dirty edge j receives the unnormalized weight

$$\omega_j^r = m_j^r \exp(-\gamma_s \bar{\tau}_j^r), \qquad \pi_{j,r} = \frac{\omega_j^r}{\sum_{\ell\in\mathcal{E}_r}\omega_\ell^r}. \tag{41}$$

The effective edge evidence and weighted cloud model are

$$n_r^{\mathrm{eff}} = \frac{1}{\sum_{j\in\mathcal{E}_r}\pi_{j,r}^2}, \qquad \widetilde{\mathbf{w}}_s^r = \sum_{j\in\mathcal{E}_r}\pi_{j,r}\mathbf{w}_j. \tag{42}$$

The server applies the same proximal-momentum principle as the edge. After the cloud step, edge j receives a soft synchronization

$$\mathbf{w}_j^+ = (1-\alpha_j)\mathbf{w}_j^- + \alpha_j \mathbf{w}_s^+, \tag{43}$$

where participating dirty edges may use a larger α_j than currently clean edges.

C Communication-Controller Details

C.1 Physical reference scales

The two timers are normalized by separate physical scales because the client–edge and edge–cloud processes have different dynamics. If S_i is the characteristic completion time of client i, define the nominal supply rate of edge j as

$$\Lambda_j = \sum_{i:e(i)=j} \frac{1}{\mathbb{E}[S_i]}. \tag{44}$$

A natural edge scale is the median time required to collect one nominal full buffer,

$$\tau_H = \mathrm{median}_j \left(\frac{B_{\max}}{\Lambda_j} \right). \tag{45}$$

If L_j denotes edge–cloud latency, the cloud scale can be chosen as

$$\tau_T = \sqrt{\mathbb{E}[L_j^2]}. \tag{46}$$

The normalized timers are restricted to positive compact intervals $x_H \in \mathcal{X}_H = [\underline{x}_H, \overline{x}_H]$ and $x_T \in \mathcal{X}_T = [\underline{x}_T, \overline{x}_T]$.

C.2 Innovation disagreement

For model contributions $\{\mathbf{w}_q\}_{q=1}^K$ with normalized weights p_q and reference model $\mathbf{w}_{\mathrm{ref}}$, define

$$\Delta_q = \mathbf{w}_q - \mathbf{w}_{\mathrm{ref}}, \qquad \bar{\Delta} = \sum_q p_q \Delta_q, \qquad n_{\mathrm{eff}} = \frac{1}{\sum_q p_q^2}. \tag{47}$$

The scale-free disagreement statistic is

$$D = \frac{\sum_q p_q \|\Delta_q - \bar{\Delta}\|_2^2}{n_{\mathrm{eff}} \|\bar{\Delta}\|_2^2 + \epsilon}. \tag{48}$$

At an edge flush this gives D_H; at a cloud event it gives D_T.

C.3 Self-normalized edge pressure

For edge flush k, let Δ_k^H be the time since the previous flush of that edge, B_k the raw number of buffered updates, B_k^{eff} the effective evidence, $a_{H,k}$ the weighted mean age, and $D_{H,k}$ the disagreement. The observed client-update supply is

$$\lambda_{H,k} = \frac{B_k}{\max(\Delta_k^H, \epsilon)}. \tag{49}$$

Let barred quantities denote exponentially smoothed baselines from previous observations, with smoothing factor β. For target evidence $n_H^\star$ (typically $B_{\max}$), define

$$A_{H,k} = \frac{\bar{\lambda}_{H,k-1}}{\max(\lambda_{H,k}, \epsilon)} \frac{n_H^\star}{\max(B_k^{\mathrm{eff}}, \epsilon)}, \tag{50}$$

$$B_{H,k} = \frac{a_{H,k}}{\max(\bar{a}_{H,k-1}, \epsilon)} \frac{1 + D_{H,k}}{1 + \bar{D}_{H,k-1}}. \tag{51}$$

Slow update supply or insufficient effective evidence therefore increases A_H and tends to lengthen the collection time, while unusually old or inconsistent updates increase B_H and shorten it.

C.4 Self-normalized cloud pressure

For cloud event r, let $\Delta_r^T = t_r - t_{r-1}$ and

$$\nu_{T,r} = \frac{1}{\max(\Delta_r^T, \epsilon)} \tag{52}$$

be the observed cloud aggregation frequency. For weighted mean age $a_{T,r}$, disagreement $D_{T,r}$, effective edge evidence n_r^{eff}, and desired evidence $n_T^\star$ (typically $Q_{\min}$), define

$$A_{T,r} = \frac{\nu_{T,r}}{\max(\bar{\nu}_{T,r-1}, \epsilon)} \frac{n_T^\star}{\max(n_r^{\text{eff}}, \epsilon)}, \tag{53}$$

$$B_{T,r} = \frac{a_{T,r}}{\max(\bar{a}_{T,r-1}, \epsilon)} \frac{1 + D_{T,r}}{1 + \bar{D}_{T,r-1}}. \tag{54}$$

Frequent cloud aggregation or weak effective edge evidence therefore increases the pressure to wait longer, while excessive age or disagreement shortens the cloud deadline.

C.5 Convex targets and online tracking

The edge and cloud targets are obtained independently as

$$x_{H,k}^\star = \Pi_{\mathcal{X}_H} \sqrt{\frac{A_{H,k}}{B_{H,k}}}, \qquad x_{T,r}^\star = \Pi_{\mathcal{X}_T} \sqrt{\frac{A_{T,r}}{B_{T,r}}}. \tag{55}$$

The corresponding physical timers are $T_{\text{hold}} = \tau_H x_H$ and $T_{\max} = \tau_T x_T$. Each target is tracked by the projected log-domain recursion equation 72 using its own gain and clipping threshold.

D Complete Convergence Analysis

This section analyzes two complementary properties of OCS-BAHFL. First, we establish that the proposed self-normalized communication controller preserves strictly positive and uniformly bounded edge and cloud waiting times and tracks its instantaneous convex timer targets in the logarithmic domain. Second, we show that these bounded communication delays preserve the standard stationary-point convergence behavior of asynchronous non-convex federated optimization, up to residual terms caused by stochastic gradients, statistical heterogeneity, local-update drift, asynchronous model age, and proximal-momentum approximation.

It is important to distinguish the two levels of analysis. The global learning objective is generally non-convex. Convexity is invoked only for the scalar communication-surrogate problems that determine T_{hold} and $T_{\max}$. The optimization convergence result therefore does not require convexity of the neural-network training objective.

D.1 Assumptions

Assumption 1 (Smooth and lower-bounded objectives). The global objective

$$F(\mathbf{w}) = \sum_{i=1}^{N} p_i F_i(\mathbf{w}) \tag{56}$$

is bounded below by some finite constant F_{inf}. Moreover, every local objective F_i is L-smooth, i.e.,

$$\|\nabla F_i(\mathbf{w}) - \nabla F_i(\mathbf{w}')\| \le L\|\mathbf{w} - \mathbf{w}'\|, \qquad \forall \mathbf{w}, \mathbf{w}'. \tag{57}$$

Assumption 2 (Unbiased stochastic gradients and bounded variance). For every client i, stochastic sample ξ, and model $\mathbf{w}$,

$$\mathbb{E}_\xi[\mathbf{g}_i(\mathbf{w}; \xi)] = \nabla F_i(\mathbf{w}), \tag{58}$$

and

$$\mathbb{E}_\xi \left[\|\mathbf{g}_i(\mathbf{w}; \xi) - \nabla F_i(\mathbf{w})\|^2\right] \le \sigma^2. \tag{59}$$

Assumption 3 (Bounded client heterogeneity). There exists $\zeta^2 \geq 0$ such that

$$\sum_{i=1}^{N} p_i \|\nabla F_i(\mathbf{w}) - \nabla F(\mathbf{w})\|^2 \leq \zeta^2, \qquad \forall \mathbf{w}. \tag{60}$$

Assumption 4 (Bounded second moment and local computation). There exists $G > 0$ such that

$$\mathbb{E}\left[\|\mathbf{g}_i(\mathbf{w};\xi)\|^2\right] \leq G^2. \tag{61}$$

The number of stochastic local-update steps performed between two successive client communications is uniformly bounded by $E_{\max} < \infty$.

Assumption 5 (Positive and bounded communication regions). The fixed parameters $B_{\max}$, $Q_{\min}$, and $T_{\min}$ are finite and positive. The normalized adaptive timers satisfy

$$x_{H,k} \in \mathcal{X}_H = [\underline{x}_H, \overline{x}_H], \qquad x_{T,r} \in \mathcal{X}_T = [\underline{x}_T, \overline{x}_T], \tag{62}$$

with

$$0 < \underline{x}_H < \overline{x}_H < \infty, \qquad 0 < \underline{x}_T < \overline{x}_T < \infty. \tag{63}$$

The physical scales τ_H and τ_T are finite and strictly positive.

Assumption 6 (Stable aggregation parameters). The client learning rate and the edge/server aggregation step sizes are chosen sufficiently small for the usual smooth non-convex descent arguments to hold. The proximal coefficients are finite and nonnegative, with at least one positive coefficient in every proximal target, and the momentum parameters satisfy

$$0 \leq \mu_e < 1, \qquad 0 \leq \mu_s < 1. \tag{64}$$

Assumption 7 (Bounded communication-observation moments). The edge- and cloud-level age and disagreement statistics used by the controller have finite second moments. In particular,

$$\mathbb{E}[a_{H,k}^2] < \infty, \qquad \mathbb{E}[a_{T,r}^2] < \infty, \tag{65}$$

and

$$\mathbb{E}[D_{H,k}^2] < \infty, \qquad \mathbb{E}[D_{T,r}^2] < \infty. \tag{66}$$

This assumption does not require the communication process to be stationary. It only excludes unbounded observation magnitudes over a finite operating regime.

D.2 Effective Evidence and Buffered Stochastic Noise

The inverse-concentration quantities used in OCS-BAHFL have a direct variance interpretation. Consider an aggregation event with normalized nonnegative weights $\{a_\ell\}_{\ell=1}^{B}$ satisfying $\sum_{\ell=1}^{B} a_\ell = 1$. Define its effective evidence as

$$B^{\mathrm{eff}} = \frac{1}{\sum_{\ell=1}^{B} a_\ell^2}. \tag{67}$$

Lemma 1 (Weighted stochastic-noise reduction)**.** *Let $\mathbf{g}_\ell$ denote stochastic gradients whose conditional stochastic errors*

$$\boldsymbol{\epsilon}_\ell = \mathbf{g}_\ell - \nabla F_{i_\ell}(\mathbf{w}_\ell) \tag{68}$$

are conditionally zero-mean and mutually independent. Then

$$\mathbb{E}\left[\left\|\sum_{\ell=1}^{B} a_\ell \boldsymbol{\epsilon}_\ell\right\|^2\right] \leq \sigma^2 \sum_{\ell=1}^{B} a_\ell^2 = \frac{\sigma^2}{B^{\mathrm{eff}}}. \tag{69}$$

Proof. Conditional zero mean and independence eliminate all cross terms, giving

$$\mathbb{E}\left[\left\|\sum_{\ell=1}^{B} a_\ell \boldsymbol{\epsilon}_\ell\right\|^2\right] = \sum_{\ell=1}^{B} a_\ell^2 \mathbb{E}[\|\boldsymbol{\epsilon}_\ell\|^2] \leq \sigma^2 \sum_{\ell=1}^{B} a_\ell^2. \tag{70}$$

Using $B^{\text{eff}} = 1/\sum_\ell a_\ell^2$ completes the result. ■

For equal weights, $B^{\text{eff}} = B$, and equation 69 reduces to the familiar σ^2/B variance reduction.

The result applies at both hierarchy levels. At an edge flush, B^{eff} corresponds to the effective client evidence B_k^{eff}. At a cloud event, the same argument with the normalized edge weights $\{\pi_{j,r}\}$ gives

$$\mathbb{E}\left[\left\|\sum_{j\in\mathcal{E}(t_r)} \pi_{j,r}\boldsymbol{\epsilon}_j\right\|^2\right] \leq \frac{\sigma^2_{\text{edge}}}{n_r^{\text{eff}}}. \tag{71}$$

Effective evidence therefore captures not merely the number of received updates but also their concentration under staleness-aware weighting.

D.3 Feasibility and Boundedness of the Adaptive Timers

The normalized edge and cloud timers are updated through projected log-domain tracking. For a generic timer x_q, where $q \in \{H, T\}$, the recursion has the form

$$x_{q,k+1} = \Pi_{\mathcal{X}_q}\left[\exp\left(\log x_{q,k} + \eta_q e_{q,k}\right)\right], \tag{72}$$

where

$$e_{q,k} = \text{clip}\left(\log \frac{x^\star_{q,k}}{x_{q,k}}, -\Delta_q, \Delta_q\right). \tag{73}$$

Lemma 2 (Positive feasible timer iterates)**.** *Under Assumption 5, if $x_{q,0} \in \mathcal{X}_q$, then*

$$x_{q,k} \in \mathcal{X}_q \tag{74}$$

for every controller event k. Consequently,

$$\underline{x}_H \tau_H \leq T_{\text{hold},k} \leq \overline{x}_H \tau_H, \tag{75}$$

and

$$\underline{x}_T \tau_T \leq T_{\max,r} \leq \overline{x}_T \tau_T. \tag{76}$$

Proof. The argument of the exponential in equation 72 is finite because $|e_{q,k}| \leq \Delta_q$. Hence the unprojected iterate is strictly positive. Projection onto the closed positive interval $\mathcal{X}_q$ guarantees

$$x_{q,k+1} \in \mathcal{X}_q. \tag{77}$$

Induction establishes feasibility for every controller event. Multiplication by the positive physical scales τ_H and τ_T gives equation 75 and equation 76. ■

Define

$$T_H^{\max} = \overline{x}_H \tau_H, \qquad T_T^{\max} = \overline{x}_T \tau_T. \tag{78}$$

These quantities provide explicit deterministic upper bounds on the adaptive waiting intervals.

D.4 Bounded Hierarchical Waiting Time

Lemma 3 (Bounded edge-buffer residence)**.** *Suppose a client update has entered a nonempty edge buffer at time t_i^{arr}. Under Assumption 5 and the edge trigger equation 2, its residence time before the associated buffer flush is bounded by*

$$T_H^{\max} = \overline{x}_H \tau_H, \tag{79}$$

up to the granularity with which communication events are processed.

Proof. The edge flush condition is satisfied when either

$$|\mathcal{B}_j| \geq B_{\max} \tag{80}$$

or

$$a_j^{\mathrm{buf}} \geq T_{\mathrm{hold}}. \tag{81}$$

From Lemma 2,

$$T_{\mathrm{hold}} \leq T_H^{\max}. \tag{82}$$

Thus, even if the buffer-size trigger never occurs, the time-based trigger must become satisfied once the oldest update has resided for at most $T_H^{\max}$. ■

Lemma 4 (Bounded cloud waiting)**.** *Suppose the dirty-edge set is nonempty after the previous cloud event. Then the deadline branch of equation 3 forces a subsequent cloud aggregation within at most*

$$T_T^{\max} = \overline{x}_T \tau_T \tag{83}$$

time units after the previous cloud event, up to event-processing granularity.

Proof. Whenever $|\mathcal{E}(t)| > 0$, the second branch of equation 3 becomes true when

$$\Delta t_s \geq T_{\max}(t). \tag{84}$$

Lemma 2 guarantees

$$T_{\max}(t) \leq T_T^{\max}. \tag{85}$$

Therefore, available dirty-edge information cannot remain pending beyond the maximum admissible cloud deadline. ■

The two lemmas imply that adaptation does not remove the bounded-delay property required by asynchronous optimization. Rather, the controller changes the delay inside a compact admissible region.

D.5 CONTROLLER TARGET TRACKING

The convex communication objectives derived in Appendix C produce instantaneous normalized targets

$$x^{\star}_{q,k} = \Pi_{\mathcal{X}_q}\left(\sqrt{\frac{A_{q,k}}{B_{q,k}}}\right), \qquad q \in \{H, T\}. \tag{86}$$

We first characterize the stationary-target case.

Lemma 5 (Log-domain contraction)**.** *Suppose $x^{\star}_q$ is constant over a sequence of controller events, projection is inactive, and the clipping threshold is inactive. Then the logarithmic tracking error*

$$\varepsilon_{q,k} = \log x_{q,k} - \log x^{\star}_q \tag{87}$$

satisfies

$$\varepsilon_{q,k+1} = (1 - \eta_q)\varepsilon_{q,k}. \tag{88}$$

Hence, for $0 < \eta_q \leq 1$,

$$|\varepsilon_{q,k}| \leq (1 - \eta_q)^k |\varepsilon_{q,0}|. \tag{89}$$

Proof. When neither projection nor clipping is active,

$$\log x_{q,k+1} = \log x_{q,k} + \eta_q \left(\log x^{\star}_q - \log x_{q,k}\right). \tag{90}$$

Subtracting $\log x^{\star}_q$ from both sides gives

$$\varepsilon_{q,k+1} = (1 - \eta_q)\varepsilon_{q,k}. \tag{91}$$

Repeated application yields equation 89. ■

The stationary-target assumption is useful for characterizing the local behavior of the controller. In practice, the target changes as communication conditions change. The following result captures this tracking behavior.

Lemma 6 (Tracking of a time-varying timer target)**.** *Suppose projection and clipping are inactive and define*

$$\delta_{q,k} = \log x^{\star}_{q,k+1} - \log x^{\star}_{q,k}. \tag{92}$$

Then

$$\varepsilon_{q,k+1} = (1 - \eta_q)\varepsilon_{q,k} - \delta_{q,k}. \tag{93}$$

If

$$|\delta_{q,k}| \leq \bar{\delta}_q \qquad \forall k, \tag{94}$$

then

$$\limsup_{k\to\infty} |\varepsilon_{q,k}| \leq \frac{\bar{\delta}_q}{\eta_q}. \tag{95}$$

Proof. From the unprojected recursion,

$$\log x_{q,k+1} = (1-\eta_q)\log x_{q,k} + \eta_q \log x^\star_{q,k}. \tag{96}$$

Subtracting $\log x^\star_{q,k+1}$ gives

$$\varepsilon_{q,k+1} = (1-\eta_q)\varepsilon_{q,k} - \delta_{q,k}. \tag{97}$$

Therefore,

$$|\varepsilon_{q,k+1}| \le (1-\eta_q)|\varepsilon_{q,k}| + \bar{\delta}_q. \tag{98}$$

Iteration of this recursion gives

$$|\varepsilon_{q,k}| \le (1-\eta_q)^k |\varepsilon_{q,0}| + \bar{\delta}_q \sum_{\ell=0}^{k-1} (1-\eta_q)^\ell, \tag{99}$$

and the geometric-series limit yields equation 95. ■

Lemma 6 formalizes the intended role of the tracking gain. Larger η_q responds more rapidly to changes in the communication regime, whereas a smaller value provides stronger smoothing of noisy asynchronous observations.

Projection and clipping further restrict individual controller moves and preserve boundedness. They do not alter the fundamental interpretation that the timer tracks a moving convex optimum.

D.6 Bound on Asynchronous Model Drift

The convergence effect of communication scheduling appears through the difference between the model at which an update was generated and the model at which it is eventually incorporated.

Let $\widehat{\tau}_k$ denote the effective end-to-end age associated with an update used at server event k. Under bounded local work and bounded communication waiting, there exists a finite constant $\tau_{\text{eff}}^{\max}$ such that

$$\widehat{\tau}_k \le \tau_{\text{eff}}^{\max}. \tag{100}$$

The bound includes client computation, client–edge communication, edge-buffer residence, edge–cloud communication, and cloud waiting. The controller-dependent contribution is bounded explicitly through

$$T_H^{\max} + T_T^{\max}. \tag{101}$$

Lemma 7 (Staleness-induced gradient discrepancy)**.** *Under Assumptions 1 and 4, suppose an update incorporated at server event k was evaluated using a model $\mathbf{w}^{k-\delta}$ whose effective age is bounded by $\tau_{\text{eff}}^{\max}$. Then there exists a constant $C_\tau > 0$, independent of K, such that*

$$\mathbb{E}\left[\|\nabla F(\mathbf{w}_s^k) - \nabla F(\mathbf{w}^{k-\delta})\|^2\right] \le C_\tau L^2 G^2 \left(\tau_{\text{eff}}^{\max}\right)^2. \tag{102}$$

Proof sketch. By L-smoothness,

$$\|\nabla F(\mathbf{w}_s^k) - \nabla F(\mathbf{w}^{k-\delta})\| \le L\|\mathbf{w}_s^k - \mathbf{w}^{k-\delta}\|. \tag{103}$$

The model difference is the accumulation of a bounded number of intermediate update steps. Assumption 4 bounds their second moments by a quantity proportional to G^2. Because the effective age is uniformly bounded by $\tau_{\text{eff}}^{\max}$, squaring the accumulated model drift yields equation 102, with fixed algorithm-dependent quantities absorbed into C_τ. ■

The key implication is that adaptation changes the magnitude of the asynchronous error but cannot make it unbounded, since both adaptive timers remain inside compact feasible regions.

D.7 Hierarchical Update Decomposition

Let $\widehat{\mathbf{g}}^{\,k}$ denote the effective descent direction represented by the hierarchical aggregation performed at server event k. We decompose it as

$$\widehat{\mathbf{g}}^{\,k} = \nabla F(\mathbf{w}_s^k) + \boldsymbol{\epsilon}_{\text{stoch}}^k + \boldsymbol{\epsilon}_{\text{het}}^k + \boldsymbol{\epsilon}_{\text{stale}}^k + \boldsymbol{\epsilon}_{\text{local}}^k + \boldsymbol{\epsilon}_{\text{pm}}^k. \tag{104}$$

The terms have the following interpretations. The stochastic term results from mini-batch sampling. The heterogeneity term results from differences among client objectives. The stale term represents evaluating or applying updates generated from older model states. The local term captures drift introduced by multiple local optimization steps before communication. The final term represents the discrepancy introduced by the hierarchical proximal-momentum transformation relative to an idealized direct descent step.

Under the preceding assumptions, these components satisfy bounds of the form

$$\mathbb{E}[\|\boldsymbol{\epsilon}_{\text{stoch}}^k\|^2] \le C_\sigma \frac{\sigma^2}{N_{\text{eff}}^k}, \tag{105}$$

$$\mathbb{E}[\|\boldsymbol{\epsilon}_{\text{het}}^k\|^2] \le C_\zeta \zeta^2, \tag{106}$$

$$\mathbb{E}[\|\boldsymbol{\epsilon}_{\text{stale}}^k\|^2] \le C_\tau L^2 G^2 \left(\tau_{\text{eff}}^{\max}\right)^2, \tag{107}$$

$$\mathbb{E}[\|\boldsymbol{\epsilon}_{\text{local}}^k\|^2] \le \mathcal{E}_{\text{local}}, \tag{108}$$

and

$$\mathbb{E}[\|\boldsymbol{\epsilon}_{\text{pm}}^k\|^2] \le \mathcal{E}_{\text{pm}}, \tag{109}$$

where all constants are independent of the number of server events K.

Here N_{eff}^k denotes the effective hierarchical evidence contributing to event k. A conservative choice is

$$N_{\text{eff}}^k = \min\left\{ \min_{j \in \mathcal{E}(t_k)} B_{j,k}^{\text{eff}},\ n_k^{\text{eff}} \right\}, \tag{110}$$

which captures the fact that concentration at either hierarchy level can limit stochastic averaging.

D.8 Stationary-Point Convergence

We now state the main optimization result.

Theorem 1 (Convergence to a first-order stationary neighborhood)**.** *Let $\{\mathbf{w}_s^k\}_{k=0}^K$ denote the sequence of global models produced by OCS-BAHFL. Under Assumptions 1–7 and sufficiently small effective optimization steps, there exist finite positive constants $C_0, \ldots, C_5$, independent of K, such that*

$$\frac{1}{K}\sum_{k=0}^{K-1}\mathbb{E}\left[\|\nabla F(\mathbf{w}_s^k)\|^2\right] \le \frac{C_0\left(F(\mathbf{w}_s^0)-F_{\inf}\right)}{\eta_s K} + C_1\eta_s\sigma^2\left(\frac{1}{K}\sum_{k=0}^{K-1}\mathbb{E}\left[\frac{1}{N_{\text{eff}}^k}\right]\right) + C_2\eta_s\zeta^2 + C_3\eta_s L^2G^2\left(\tau_{\text{eff}}^{\max}\right)^2 + C_4\mathcal{E}_{\text{local}} + C_5\mathcal{E}_{\text{pm}}. \tag{111}$$

Therefore, the adaptive communication controller preserves convergence to a bounded neighborhood of a first-order stationary point.

Proof sketch. By L-smoothness,

$$F(\mathbf{w}_s^{k+1}) \le F(\mathbf{w}_s^k) + \left\langle \nabla F(\mathbf{w}_s^k), \mathbf{w}_s^{k+1}-\mathbf{w}_s^k\right\rangle + \frac{L}{2}\|\mathbf{w}_s^{k+1}-\mathbf{w}_s^k\|^2. \tag{112}$$

Represent the effective server update in the generic form

$$\mathbf{w}_s^{k+1} = \mathbf{w}_s^k - \eta_s\widehat{\mathbf{g}}^{\,k}, \tag{113}$$

where fixed coefficients induced by proximal aggregation and server momentum can be absorbed into the effective step size and constants in the bound under Assumption 6.

Substitution into equation 112 gives

$$F(\mathbf{w}_s^{k+1}) \le F(\mathbf{w}_s^k) - \eta_s\left\langle \nabla F(\mathbf{w}_s^k), \widehat{\mathbf{g}}^{\,k}\right\rangle + \frac{L\eta_s^2}{2}\|\widehat{\mathbf{g}}^{\,k}\|^2. \tag{114}$$

Using equation 104, write

$$\widehat{\mathbf{g}}^{\,k} = \nabla F(\mathbf{w}_s^k) + \boldsymbol{\varepsilon}^k, \tag{115}$$

where

$$\boldsymbol{\varepsilon}^k = \boldsymbol{\epsilon}_{\text{stoch}}^k + \boldsymbol{\epsilon}_{\text{het}}^k + \boldsymbol{\epsilon}_{\text{stale}}^k + \boldsymbol{\epsilon}_{\text{local}}^k + \boldsymbol{\epsilon}_{\text{pm}}^k. \tag{116}$$

Applying Young's inequality to the inner-product cross term gives, for an appropriate constant $c > 0$,

$$-\left\langle \nabla F(\mathbf{w}_s^k), \boldsymbol{\varepsilon}^k\right\rangle \le c\|\nabla F(\mathbf{w}_s^k)\|^2 + \frac{1}{4c}\|\boldsymbol{\varepsilon}^k\|^2. \tag{117}$$

For sufficiently small η_s, the resulting coefficient multiplying $\|\nabla F(\mathbf{w}_s^k)\|^2$ remains negative in the expected descent inequality. The error decomposition and equation 105–equation 109 then yield

$$\mathbb{E}[F(\mathbf{w}_s^{k+1})] \le \mathbb{E}[F(\mathbf{w}_s^k)] - c_0\eta_s\mathbb{E}\left[\|\nabla F(\mathbf{w}_s^k)\|^2\right] + c_1\eta_s^2\frac{\sigma^2}{N_{\text{eff}}^k} + c_2\eta_s^2\zeta^2 + c_3\eta_s^2L^2G^2\left(\tau_{\text{eff}}^{\max}\right)^2 + c_4\eta_s\mathcal{E}_{\text{local}} + c_5\eta_s\mathcal{E}_{\text{pm}}, \tag{118}$$

for finite positive constants $c_0, \ldots, c_5$.

Summing equation 118 from $k = 0$ to $K - 1$ causes the objective terms to telescope. Since $F(\mathbf{w}_s^K) \geq F_{\inf}$, division by $K\eta_s$ yields equation 111. ■

D.9 Effect of the Adaptive Communication Variables

Theorem 1 makes explicit how the two communication variables affect optimization.

The edge holding time controls the tradeoff between effective client evidence and client-update age. Increasing T_{hold} may increase B_k^{eff}, thereby reducing the stochastic term

$$\frac{\sigma^2}{N_{\text{eff}}^k}, \tag{119}$$

but excessive holding can increase the age component of $\tau_{\text{eff}}^{\max}$.

Similarly, $T_{\max}$ controls the amount of time available for dirty-edge evidence to accumulate before cloud aggregation. A longer deadline may increase effective edge participation n_k^{eff} and reduce communication frequency, but excessive cloud waiting increases the staleness component in equation 107.

The convex timer objectives therefore have the same qualitative tradeoff as the stationary-point bound: communication/evidence pressure favors additional accumulation time, while age and disagreement pressure favor earlier propagation.

Importantly, this interpretation does not imply that either timer must increase or decrease monotonically during training. The controller responds to the observed asynchronous operating state. If effective evidence is insufficient and updates remain fresh, additional waiting can be beneficial. If evidence is already sufficient while update age or innovation disagreement increases, a shorter interval is preferred.

D.10 Asymptotic Consequences

Corollary 1 (Constant effective step size)**.** *For a sufficiently small constant effective server step size η_s, OCS-BAHFL converges to a first-order stationary neighborhood whose radius is controlled by*

$$\mathcal{O}\left(\eta_s \frac{\sigma^2}{N_{\text{eff}}} + \eta_s \zeta^2 + \eta_s L^2 G^2 (\tau_{\text{eff}}^{\max})^2 + \mathcal{E}_{\text{local}} + \mathcal{E}_{\text{pm}}\right), \tag{120}$$

where N_{eff} denotes a representative lower effective-evidence level.

Corollary 2 (Finite-horizon diminishing step size)**.** *If the effective server step size is selected as*

$$\eta_s = \Theta(K^{-1/2}), \tag{121}$$

then the optimization-dependent terms satisfy

$$\frac{F(\mathbf{w}_s^0) - F_{\inf}}{\eta_s K} = \mathcal{O}(K^{-1/2}), \tag{122}$$

and

$$\eta_s \left[\frac{\sigma^2}{N_{\text{eff}}} + \zeta^2 + L^2 G^2 (\tau_{\text{eff}}^{\max})^2\right] = \mathcal{O}(K^{-1/2}) \tag{123}$$

when the quantities inside the brackets remain uniformly bounded.

Thus,

$$\frac{1}{K}\sum_{k=0}^{K-1}\mathbb{E}\left[\|\nabla F(\mathbf{w}_s^k)\|^2\right] = \mathcal{O}(K^{-1/2}) + \mathcal{O}(\mathcal{E}_{\mathrm{local}}) + \mathcal{O}(\mathcal{E}_{\mathrm{pm}}). \tag{124}$$

If the local-update and proximal-momentum approximation errors also vanish under the chosen optimization schedule, the standard first-order stationarity condition

$$\lim_{K\to\infty}\frac{1}{K}\sum_{k=0}^{K-1}\mathbb{E}\left[\|\nabla F(\mathbf{w}_s^k)\|^2\right] = 0 \tag{125}$$

is recovered.

D.11 Discussion

The preceding analysis establishes three properties relevant to the proposed communication scheduler.

First, the projected log-domain recursions guarantee that both adaptive communication variables remain strictly positive and uniformly bounded. Consequently, adapting T_{hold} and $T_{\max}$ cannot introduce unbounded buffering or cloud waiting.

Second, for a stationary communication regime, each timer contracts geometrically toward the unique minimizer of its strictly convex scalar surrogate. For slowly varying regimes, the timer instead tracks a neighborhood of the moving optimum whose size depends on the rate at which the communication state changes relative to the controller gain.

Third, bounded adaptive communication delays enter the learning analysis only through bounded asynchronous-drift terms. Hence, the proposed controller does not change the qualitative stationary-point convergence class of asynchronous non-convex FL. Rather, it dynamically modifies the constants in the convergence neighborhood by balancing effective aggregation evidence against staleness and update disagreement.

The controller therefore does not rely on validation accuracy to decide whether communication should become more or less frequent. Its effect on optimization is mediated entirely through communication-side quantities already generated during asynchronous training: effective evidence, arrival or aggregation rate, model age, and innovation disagreement.

E Compact Notation Summary

Table 2: Main notation used in OCS-BAHFL.

Symbol	Meaning
$\mathcal{B}_j$	Client-update buffer at edge j
$\mathcal{E}$	Dirty-edge set
$B_{\max}$	Edge-buffer capacity
$Q_{\min}$	Cloud quorum
T_{hard}	Fixed cloud liveness deadline
T_{hold}	Adaptive edge holding time
$T_{\max}$	Adaptive cloud deadline
B^{eff}	Effective client evidence at an edge flush
n^{eff}	Effective edge evidence at a cloud event
D_H, D_T	Edge/cloud innovation disagreement
τ_H, τ_T	Edge/cloud physical reference scales
x_H, x_T	Normalized adaptive timers