VALUING UPDATES, NOT ROUNDS, IN FULLY ASYNCHRONOUS FEDERATED LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

Contribution evaluation assigns credit to clients in federated learning and underpins incentives and participation. Existing methods rely on synchronous rounds, which supply fixed participants, a common reference model, and a well-defined aggregate for every subset of updates. Fully asynchronous training removes all three. Replayed on the same training traces, eight published evaluators rank clients in conflicting ways, so it is unclear what quantity their scores estimate. We introduce FedACE, which values each asynchronous update through its own cooperative game. The players are the update itself and the updates applied between its dispatch and its arrival, the reference is the global model from which it was computed, and every coalition is composed in the recorded arrival order. All three are read from the execution record rather than chosen by the evaluator. We prove that this game is well-posed, that the recorded sequence of global models identifies its composition order under an explicit non-degeneracy condition, and that it coincides with the classical round game when coalitions are composed by a synchronous batch rule. FedACE solves small games exactly by enumeration, estimates larger ones with score-blind random permutation rings, credits each update exactly once, and never alters the training trajectory. Because the game admits exact evaluation, we score every estimator against the same exact Shapley value rather than a proxy. Across 27,000 updates on CIFAR-10, CIFAR-100, and Tiny-ImageNet, FedACE solves 73.4% of the games exactly. On all 681 nine-update games, it uses 23% of the utility queries required by exhaustive enumeration and is closer to the exact value than every compared estimator in each of the nine training runs. At the same query budget, Monte Carlo sampling incurs 1.7 to 1.9 $\times$ the error of FedACE.

1 INTRODUCTION

Federated learning (FL) (McMahan et al., 2017) lets clients train a shared global model while keeping raw data decentralized. Once model ownership is separated from data ownership, client contribution becomes consequential for incentive allocation, participation, and accountability (Wu et al., 2024). Most contribution-evaluation methods assume round-based FL (Ghorbani & Zou, 2019). In a standard synchronous round, the server dispatches one global model to a selected client set, waits for their local training, and aggregates a frozen collection of uploads through an order-invariant rule (McMahan et al., 2017; Li et al., 2020). The round therefore supplies a participant set, a common reference state, and enough retained uploads to reconstruct the aggregate of any subset. Because every round waits for its slowest member, stragglers set the pace of training (Kairouz & McMahan, 2021), and fully asynchronous FL removes this barrier (Xie et al., 2020). In the asynchronous protocol studied here, a client trains from the model version available at dispatch, returns whenever it finishes, and is mixed into the current server state immediately with a staleness-dependent weight (Liu et al., 2024; Xu et al., 2024). Staleness counts the updates the server applied while the client trained (Koloskova et al., 2022). Updates need not share a starting version. Across multi-benchmark traces, 78.9% of arrivals occur after at least one intervening update, and persistent stragglers exhibit substantially larger staleness than fast clients (Yang et al., 2021) (Fig. 1a). The round is gone, but contribution scores can still determine compensation and future participation (Xu et al., 2021; Wu et al., 2024). A ranking is therefore produced even when the quantity being ranked has not been made explicit (Chen et al., 2023; Fan et al., 2024).

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.

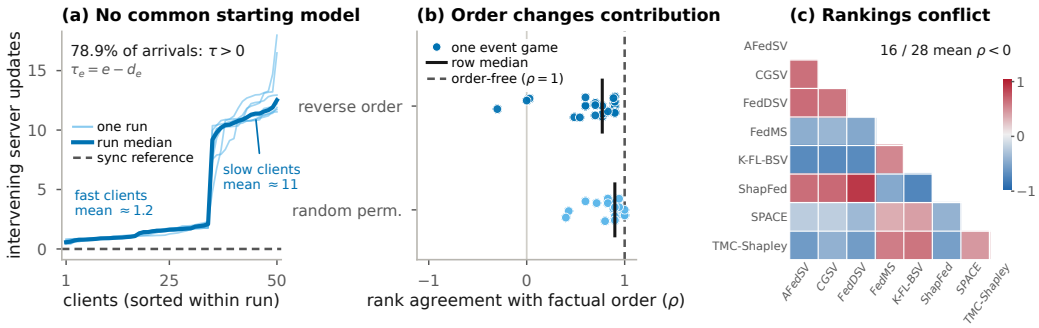

Figure 1: **Fully asynchronous training removes the shared round that contribution evaluation relies on.** (a) Clients no longer start from a common model, since 78.9% of updates arrive after the server has already changed, and about 11 server updates happen while a slow client trains against 1.2 for a fast one. (b) Applying the same updates in a different order changes how they are ranked, so the order must be part of the definition of contribution. (c) Eight published evaluators, compared only with each other on the same training runs, disagree on how to rank clients, and 16 of their 28 pairwise rank correlations are negative.

Existing contribution evaluation in synchronous federated learning (SFL) predominantly relies on the Shapley value (SV) (Shapley, 1953), which is defined for a finite cooperative game (N, v) with a player set N and a function v that assigns a scalar to every coalition $S \subseteq N$. Within a synchronous round, clients naturally serve as players, and aggregating any subset of frozen uploads supplies v, so the synchronous literature (Ghorbani & Zou, 2019; Chen et al., 2023; Tastan et al., 2024; Yang et al., 2025) can focus on reducing the cost of coalition queries. Fully asynchronous execution removes this default construction. Without a round boundary, the execution does not identify which update events should participate in the value of a newly completed update. Updates are sequential operators, so a subset alone does not determine a candidate state, and changing only their composition order changes the resulting event-value rankings (Fig. 1b). The server also observes a single factual trajectory, whereas coalition valuation asks for states in which selected updates were omitted. Published methods capture useful aspects of contribution through geometric alignment, validation gains, data valuation, or aggregation-aware proxies (Xu et al., 2021; Zhang et al., 2026; Jiang et al., 2023; Lei et al., 2025; Tastan et al., 2024; Chen et al., 2023; Yang et al., 2025; Ghorbani & Zou, 2019), but their scores need not estimate one shared asynchronous quantity, and on identical traces 16 of 28 pairwise mean client-ranking correlations are negative (Fig. 1c). This conflict does not by itself invalidate any proxy, and it shows that approximation error is not interpretable until the participants, coalition composition, and counterfactual protocol are declared.

Our key observation is that each completed update carries a dependency interval defined by the execution itself. Consider event e, whose client trained from server version d_e and returned at event e. We take as players the dependency interval $C_e = \{d_e, \ldots, e\}$, which records the updates applied after dispatch and before the returned update is applied, together with e itself. For a coalition $S \subseteq C_e$, evaluation starts from the recorded dispatch reference, keeps the factual local models and mixing coefficients of the retained events, omits events outside S, and composes the retained events in factual arrival order. This frozen-update deletion protocol maps every subset to one candidate state without pretending to simulate a complete retraining run in which a client never participated. The construction never looks ahead, because event e uses only records available when it completes. When uploads share a reference state and coalitions are composed by a synchronous batch rule, it coincides with the familiar frozen-upload round game. It therefore contains the classical game and extends it to the sequential asynchronous regime covered by our recorded-execution and valuation-invariance assumptions.

Based on this construction, we introduce **Fed**erated **A**synchronous **C**ontribution **E**valuation (FedACE), a framework for update-event Shapley valuation under fully asynchronous FL. A Dependency-Interval Game Constructor reads the reference state and the dependency interval

of each completed update. An Order-Preserving Isolated Utility Probe composes coalitions in factual order, gives each coalition its own evaluation state, and restores task parameters, buffers, module modes, and random-number-generator states after probing. A Size-Routed Shapley Solver enumerates all coalitions for $n \leq 6$ and otherwise uses two score-blind randomized rings, with exact audits excluded from routing and credit. An Anchor-Only Auditable Ledger credits only the newly completed event from each overlapping game and aggregates its event values into client scores. Execution supplies the interval, order, local models, and mixing coefficients. The evaluator declares the utility, coalition-state protocol, solution concept, and enumeration budget.

Our contributions are threefold.

- We define an event-level cooperative game on the dependency interval of each update, whose events, reference state, and composition order are read from the recorded execution. We prove that it is well posed, that the recorded prefixes identify its factual composition order under an explicit non-degeneracy condition, and that it coincides with the classical round game when coalitions are composed by a synchronous batch rule.
- FedACE solves this game with four modules. It builds the interval, probes every coalition in factual order in isolation, routes each game by its size alone to exact enumeration or to unbiased and exactly efficient score-blind rings, and credits each arrival exactly once, while the recorded training trajectory stays bit identical.
- On 27,000 arrivals from CIFAR-10, CIFAR-100, and Tiny-ImageNet, FedACE solves 73.4% of the games exactly. On all 681 games with nine events it is closer to the exact Shapley value than every compared estimator in all nine training runs while using 23% of the queries of exhaustive enumeration, and eight published evaluators replayed on the same traces are not interchangeable estimates of one asynchronous quantity.

2 Related Work

Contribution evaluation from synchronous to asynchronous FL. The Shapley value (SV) is the default instrument for client valuation because its axioms uniquely determine a division of the joint utility, and in synchronous FL its main difficulty is computational. One line of work therefore cheapens the estimate through truncated Monte Carlo sampling (Ghorbani & Zou, 2019), guided truncation with gradient-based reconstruction (Liu et al., 2022), or leave-one-out and influence-style surrogates, and another reuses the score to set aggregation weights or rewards (Xu et al., 2021; Tastan et al., 2024). Both rest on the round, which fixes the participants, averages any subset independently of client order, and confines every counterfactual removal to one round, so coupling scores with aggregation is benign there and the underlying game is rarely questioned. Asynchronous FL merges each update on arrival and damps it by staleness (Xie et al., 2020; Nguyen et al., 2022), and this damping, studied only as a device for convergence, is exactly what makes the aggregate order dependent. Asynchronous evaluators recover participants by restoring a round-shaped unit, either the first K arrivals played as a sequential cooperative game (Jiang et al., 2023; Zhang et al., 2026) or a sub-round formed after the fact (Lei et al., 2025), and score a subset by gradient or cosine consistency with the global model rather than by a declared utility. These methods then return the score to the aggregation weight, so the measured quantity depends on its own use. To our knowledge, no existing method derives its participants from the execution itself, keeps the characteristic function a set function under non-commuting composition, and confines removal to a single run.

3 Method

3.1 Execution model and records

We index arrival events by the order in which the server applies them, $e = 0, 1, 2, \ldots$, and write $c(e)$ for the client whose update arrives at event e. The global model after t applied updates is $\theta_t \in \mathbb{R}^p$, where p is the number of task parameters. The task of event e was dispatched when d_e updates had been applied, so client $c(e)$ trained from θ_{d_e}, and $\tau_e = e - d_e$ is the staleness of its update. With $\ell_e \in \mathbb{R}^p$ the returned local model, the server performs

$$\theta_{e+1} \;=\; T_e(\theta_e), \qquad T_e(\theta) \;=\; (1-\alpha_e)\,\theta \;+\; \alpha_e\,\ell_e, \qquad \alpha_e \;=\; \lambda\, w(\tau_e), \tag{1}$$

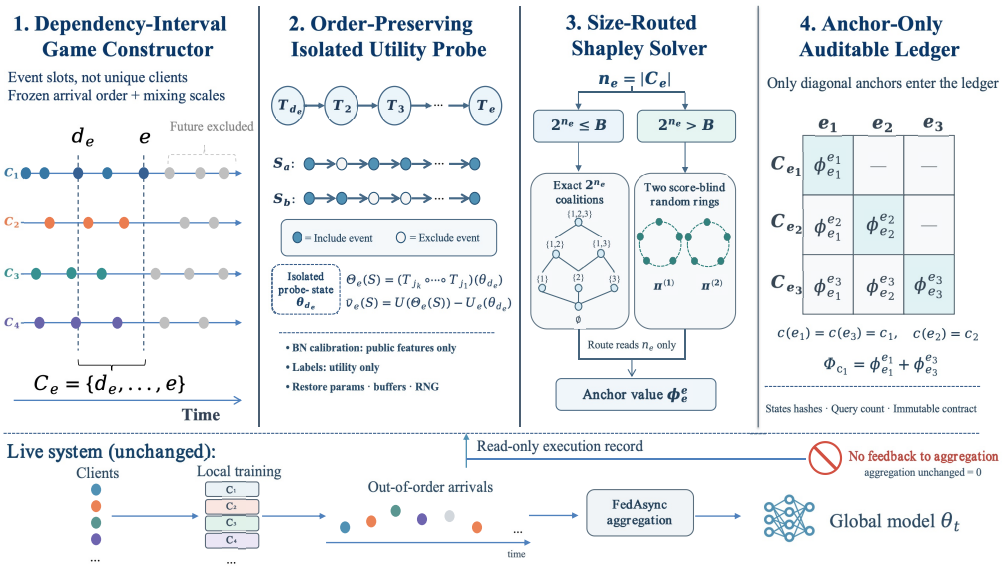

Figure 2: **FedACE values every asynchronous update without touching training.** The live system (bottom) trains and aggregates as usual and only hands a read-only record to FedACE (top). (1) For each arriving update, the constructor collects the updates the server applied while that client was training. (2) The probe replays any subset of them in the order they actually arrived, on a copy of the model, and measures how much it improves the model. (3) The solver computes the exact Shapley value when the game is small and a fast unbiased estimate otherwise. (4) The ledger credits each update once, to the client that sent it.

where T_e is the update operator of event e acting on an arbitrary state θ, $\alpha_e \in [0, 1]$ is its mixing coefficient, $\lambda \in (0, 1]$ is the base mixing rate, and w is the staleness discount, for example the hinge of FedAsync (Xie et al., 2020). The operators of different arrivals do not commute.

Assumption 1 (Recorded execution)**.** The trajectory is fixed. For every event j whose dispatch is still in flight, the server retains d_j, θ_{d_j}, ℓ_j and the applied α_j, and it discards events older than the oldest in-flight dispatch.

Assumption 2 (Valuation-invariant aggregation)**.** The mixing coefficient α_e is a function of τ_e alone and does not depend on any contribution score, which fails exactly for evaluators that return a score to the aggregation weight.

3.2 Dependency-Interval Game Constructor

Asynchrony removes the round, so the participants of a game can no longer be read off a communication boundary. The dispatch record remains, and it delimits the updates applied while a client trained.

Definition 1 (Dependency interval)**.** The dependency interval of arrival event e is $C_e = \{d_e, d_e + 1, \ldots, e\}$, the events applied from the dispatch of e up to and including e itself, and its size is $n_e = |C_e| = \tau_e + 1$. These events are the players of the cooperative game of event e, and e is its anchor.

Conditioned on the recorded dispatch reference and the frozen-update protocol, C_e is the player set the execution defines. The reference is θ_{d_e}, the state from which ℓ_e was computed, as the dispatched state is for a round. An event applied before dispatch is contained in that state and entered ℓ_e, so it cannot be removed without retraining, and an event after e has not yet occurred. Every event in between can be removed with ℓ_e unchanged, and composing all of them in factual order returns the recorded state θ_{e+1} (Proposition 2). C_e is therefore the largest set removable without retraining ℓ_e, outside degenerate configurations each of its events is needed to reach the recorded state, and Definition 1 needs no window length or hyperparameter.

The valued unit is an arrival event, so a client that completes twice inside an interval, as at 14.4% of our arrivals, holds two slots. The constructor emits read-only copies of C_e, θ_{d_e}, the pairs (ℓ_j, α_j) for $j \in C_e$, and the factual arrival order on C_e, a total order written $\prec$.

3.3 Order-preserving isolated utility probe

The Shapley value needs a value for every subset of C_e, and because the operators T_j do not commute, a subset determines a state only once an order is fixed. Permutations stay the accounting device for averaging marginals, while models are composed in the order in which the arrivals occurred.

Definition 2 (Order-anchored characteristic function)**.** For $S \subseteq C_e$ with $\prec$-increasing enumeration $j_1 \prec j_2 \prec \cdots \prec j_k$, define

$$\Theta_e(S) \;=\; \big(T_{j_k} \circ \cdots \circ T_{j_1}\big)(\theta_{d_e}), \qquad \bar{v}_e(S) \;=\; U_e\big(\Theta_e(S)\big) - U_e\big(\theta_{d_e}\big), \tag{2}$$

where each T_j uses its recorded ℓ_j and α_j, and $U_e : \mathbb{R}^p \to \mathbb{R}$ is the declared utility of event e, shared by every subset of the game.

The coalition state $\Theta_e(S)$ is the model the server would hold had it applied only the updates in S, with every local computation held at its recorded value. The evaluator declares two further choices, the utility U_e, a negative cross-entropy on a server-held public set, and a coalition-state protocol that rebuilds the batch-normalisation statistics of $\Theta_e(S)$ from public inputs only, so labels enter only through U_e.

Definition 3 (Isolated replay)**.** For $S \subseteq C_e$ the probe evaluates $U_e(\Theta_e(S))$ in an evaluation state owned by the coalition. The task parameters, buffers, module modes and random-number-generator states of the live run are restored and verified by hash, so no query alters the run it observes.

3.4 What the event game guarantees

Theorem 1 (Shapley value of the event game)**.** *$\bar{v}_e$ is a function $2^{C_e} \to \mathbb{R}$ with $\bar{v}_e(\varnothing) = 0$, so $(C_e, \bar{v}_e)$ is a finite cooperative game. Consequently a unique value on C_e satisfies efficiency, symmetry, the null player property and linearity, and it assigns to each $i \in C_e$*

$$\phi_i^e \;=\; \sum_{S \subseteq C_e \setminus \{i\}} \frac{|S|!\,(n_e - |S| - 1)!}{n_e!} \big[\bar{v}_e(S \cup \{i\}) - \bar{v}_e(S)\big]. \tag{3}$$

Theorem 1 weakens no axiom, since a set function needs one value per subset, which the fixed order $\prec$ supplies. We write $\phi^e = (\phi_i^e)_{i \in C_e}$, call ϕ_e^e the anchor value of e, and treat the Shapley value as a declared choice (Section 4.5).

Proposition 1 (Synchronous compatibility)**.** *If every update in C_e is computed from one dispatched state and coalition states are composed by the round's batch rule, which averages any subset with fixed weights and leaves the weight of absent clients on the reference state, then C_e is the round, $\Theta_e(S)$ does not depend on the order, and ϕ^e is the classical Shapley value of the round game.*

The classical game is thus the batch-rule instance of the same construction. Under sequential mixing every total order on C_e gives a well-posed game, and the next two results single out the factual one.

Proposition 2 (Factual-prefix reconstruction)**.** *Under the factual order $\prec$,*

$$\Theta_e\big(\{d_e, \ldots, k\}\big) \;=\; \theta_{k+1} \qquad \textit{for every } d_e \le k \le e, \tag{4}$$

so every prefix reproduces a state the system actually occupied, and the full interval reproduces the recorded state θ_{e+1}.

Proposition 3 (The recorded prefixes identify the factual order)**.** *Suppose every $\alpha_j > 0$ and no returned model ℓ_k equals the effective aggregate of the updates it would overtake (closed form in Appendix A.4). Then $\prec$ is the only total order on C_e whose composites satisfy Eq. 4.*

The excluded configurations form a finite union of affine subspaces of codimension p, and the condition is sharp (Appendix A.4). Factual order is thus the only order consistent with the record, which the replay checks at every arrival (Section 4.2).

Corollary 1 (Efficiency is a conservation law). $\sum_{i \in C_e} \phi_i^e = U_e(\theta_{e+1}) - U_e(\theta_{d_e})$, *the utility change the deployment underwent across the dependency interval of event* e.

Proposition 4 (Aggregation-level attribution). *Under Assumption 2 and the frozen-record protocol, which keeps the recorded staleness and mixing coefficient of every retained event,* $\bar{v}_e(S \cup \{i\}) - \bar{v}_e(S)$ *is the utility difference between the states that the server's aggregation rule produces from* $S \cup \{i\}$ *and from* S *with those coefficients. If instead the mixing coefficient depends on a contribution score, the recorded coefficients were produced in the presence of* i *and in general differ from those the rule would set in its absence, so the recorded coefficients alone do not recover that difference.*

3.5 Size-routed Shapley solver

Eq. 3 needs all 2^{n_e} coalition values, and n_e varies with the latency of each arrival. Let B be the enumeration budget, the largest number of coalitions the solver evaluates exhaustively for one game. If $2^{n_e} \leq B$ the solver evaluates Eq. 3 exactly. Otherwise it draws r rings $\pi^{(1)}, \ldots, \pi^{(r)}$, each a uniformly random ordering of C_e generated from a fixed sampler seed and never from a utility value. Each ring yields $2n_e$ paths, namely its n_e rotations and their reversals, where a path is a permutation of C_e. With $\mathcal{P}$ the resulting multiset of $2rn_e$ paths, the solver returns

$$\hat{\phi}_i^e \;=\; \frac{1}{|\mathcal{P}|} \sum_{\sigma \in \mathcal{P}} \big[\bar{v}_e(P_i^\sigma \cup \{i\}) - \bar{v}_e(P_i^\sigma) \big], \tag{5}$$

where P_i^σ is the set of events that precede i in path σ. A rotation or a reversal of a uniformly random ordering is again uniformly random, so $\hat{\phi}_i^e$ is unbiased for ϕ_i^e, and the marginals along every path telescope to $\bar{v}_e(C_e)$, so the estimate is exactly efficient. A ring thus combines position stratification with antithetic pairing (Mitchell et al., 2022).

Proposition 5 (Query complexity and routing). *Write* $n = n_e$. *Exact evaluation uses* 2^n *distinct utility queries, and Eq. 5 uses at most* $2 + r\,n(n-1)$. *For* $n \leq 3$ *the ring paths contain every permutation of* C_e *equally often, so Eq. 5 equals Eq. 3. The routing rule reads only* n, *so it consults no utility, no reference value and no label of any client.*

With $B = 64$ and $r = 2$, every game with $n_e \leq 6$ is solved exactly and a game with $n_e = 9$ costs at most 146 queries instead of 512. Exact tables built for evaluation never enter routing or credit.

3.6 Anchor-only auditable ledger

Dependency intervals overlap, so an event takes part in the game of every later arrival whose interval covers it. The ledger credits each event only in its own game. For a client c the ledger Φ_c and the reported score $\bar{\Phi}_c$ are

$$\Phi_c \;=\; \sum_{e\,:\,c(e)=c} \phi_e^e, \qquad \bar{\Phi}_c \;=\; \frac{\Phi_c}{|\{e : c(e) = c\}|}, \tag{6}$$

with ϕ_e^e replaced by $\hat{\phi}_e^e$ when the game is routed to rings. Averaging keeps clients that participate more often from being ranked ahead of clients that participate well. Efficiency holds per game, so Φ_c records the effects of a client's own updates, not a share of a global budget.

Proposition 6 (Single-entry accounting). *Each arrival event is charged once, as the anchor of its own game, and contributes to no other entry of Eq. 6. Overlapping dependency intervals therefore produce no duplicated credit, and* Φ_c *is well defined.*

The values ϕ_j^e with $j \neq e$ only define the marginal of the anchor. Because every query is served by Definition 3, a hash chain over the dispatched, returned and resulting models certifies that on the recorded event sequence the live parameters are bit identical to those of a run without valuation, and each game records its query count and configuration so every value can be recomputed.

4 Experiments

4.1 Experimental setup

We use CIFAR-10 and CIFAR-100 (Krizhevsky et al., 2009) and Tiny-ImageNet (Le et al., 2015), each partitioned over 50 clients by a Dirichlet distribution with concentration 0.5, and clients train

Table 1: Fidelity to the exact value on every game with a complete coalition table, as mean and standard deviation over three seeds. Same Q marks methods matched to the per-game budget of FedACE, realising 118 unique queries against 120, an asterisk marks a sampler of an asynchronous evaluator, bold marks the best mean, and a dagger marks a FedACE entry that is best in each run.

		$D_\pm$ to exact ↓			Spearman ρ ↑		
Method	Q	C-10	C-100	Tiny	C-10	C-100	Tiny
$n = 5$, exhaustive route, 736 games, exact enumeration takes 32 queries							
LOO	6	$.405_{\pm.027}$	$.396_{\pm.015}$	$.384_{\pm.036}$	$.593_{\pm.036}$	$.663_{\pm.035}$	$.704_{\pm.041}$
cyclic-reverse	22	$.064_{\pm.007}$	$.053_{\pm.004}$	$.053_{\pm.004}$	$.952_{\pm.009}$	$.962_{\pm.009}$	$.960_{\pm.011}$
MC-$4n$	29	$.096_{\pm.008}$	$.085_{\pm.005}$	$.087_{\pm.001}$	$.918_{\pm.006}$	$.918_{\pm.006}$	$.922_{\pm.003}$
GTG-Shapley	21	$.234_{\pm.020}$	$.239_{\pm.010}$	$.234_{\pm.031}$	$.785_{\pm.021}$	$.807_{\pm.036}$	$.834_{\pm.026}$
K-FL-BSV sampler*	17	$.183_{\pm.014}$	$.165_{\pm.007}$	$.168_{\pm.001}$	$.817_{\pm.011}$	$.821_{\pm.008}$	$.839_{\pm.005}$
AFedSV sampler*	29	$.095_{\pm.009}$	$.085_{\pm.005}$	$.087_{\pm.002}$	$.919_{\pm.008}$	$.919_{\pm.001}$	$.922_{\pm.003}$
FedDSV sampler*	32	$.096_{\pm.009}$	$.084_{\pm.005}$	$.086_{\pm.001}$	$.917_{\pm.008}$	$.920_{\pm.005}$	$.922_{\pm.004}$
FedACE	32	**0**	**0**	**0**	**1**	**1**	**1**
$n = 9$, two score-blind rings, 681 games, exact enumeration takes 512 queries							
LOO	10	$.441_{\pm.016}$	$.387_{\pm.012}$	$.355_{\pm.008}$	$.530_{\pm.023}$	$.759_{\pm.017}$	$.809_{\pm.021}$
cyclic-reverse	74	$.106_{\pm.001}$	$.087_{\pm.002}$	$.078_{\pm.003}$	$.919_{\pm.009}$	$.941_{\pm.008}$	$.952_{\pm.004}$
MC-$4n$	187	$.084_{\pm.003}$	$.069_{\pm.001}$	$.062_{\pm.002}$	$.948_{\pm.003}$	$.952_{\pm.006}$	$.960_{\pm.004}$
MC *(same Q)*	118	$.114_{\pm.004}$	$.094_{\pm.002}$	$.084_{\pm.002}$	$.920_{\pm.004}$	$.929_{\pm.010}$	$.943_{\pm.004}$
GTG-Shapley	103	$.302_{\pm.011}$	$.250_{\pm.024}$	$.227_{\pm.015}$	$.748_{\pm.028}$	$.858_{\pm.013}$	$.872_{\pm.016}$
K-FL-BSV sampler*	39	$.215_{\pm.005}$	$.176_{\pm.004}$	$.161_{\pm.005}$	$.802_{\pm.004}$	$.845_{\pm.017}$	$.874_{\pm.006}$
AFedSV sampler*	122	$.112_{\pm.004}$	$.092_{\pm.001}$	$.082_{\pm.002}$	$.923_{\pm.005}$	$.931_{\pm.009}$	$.943_{\pm.006}$
FedDSV sampler*	510	$.111_{\pm.003}$	$.093_{\pm.002}$	$.083_{\pm.003}$	$.924_{\pm.004}$	$.929_{\pm.008}$	$.945_{\pm.004}$
FedACE	120	$\mathbf{.068}^{\dagger}_{\pm.008}$	$\mathbf{.051}^{\dagger}_{\pm.003}$	$\mathbf{.046}^{\dagger}_{\pm.003}$	$\mathbf{.956}^{\dagger}_{\pm.004}$	$\mathbf{.963}^{\dagger}_{\pm.010}$	$\mathbf{.973}^{\dagger}_{\pm.002}$

a ResNet-20 (He et al., 2016) for two local epochs with batch size 64 and learning rate 0.03. The server mixes each returned model by FedAsync (Xie et al., 2020) with $\lambda = 0.6$ and a hinge staleness discount, and 16 clients are persistent stragglers. Three runs of 3,000 updates per dataset yield 27,000 event games, each recorded once on a synthetic clock and replayed identically by every evaluator, and the utility is the negative cross-entropy on 200 labelled public samples that no client holds (Appendix B.1). Every baseline estimates the declared game from the same coalition tables. They are leave-one-out (LOO), a fixed ring over arrival order (cyclic-reverse), Monte Carlo (MC) (Castro et al., 2009) and truncated Monte Carlo (TMC) (Ghorbani & Zou, 2019) with $4n$ permutations or the budget of FedACE, GTG-Shapley (Liu et al., 2022), and the samplers of K-FL-BSV (Jiang et al., 2023), AFedSV (Zhang et al., 2026) and FedDSV (Lei et al., 2025), each averaged over 50 repeats. Exact references come from complete coalition tables for the 19,807 exhaustively routed games and all 681 games with $n = 9$, and we report $D_\pm = \frac{1}{2}\left\|\hat{\phi}^e/\|\hat{\phi}^e\|_1 - \phi^e/\|\phi^e\|_1\right\|_1 \in [0, 1]$, the within-game Spearman correlation ρ and the number Q of unique utility queries, with each training run as one statistical unit.

The remaining experiments answer four questions. Section 4.2 measures accuracy and cost against the exact value, Section 4.3 checks its guarantees at every arrival, Section 4.4 isolates each module, and Section 4.5 varies the declared choices and locates the published evaluators.

4.2 Exact when affordable, closest otherwise

On every audited game size, FedACE is the most accurate estimator at every compared budget. At $n = 5$ it returns the exact value, while every cheaper estimator leaves a gap between 0.053 and 0.405 (Table 1). At $n = 9$ it spends 120 queries, 23% of exhaustive enumeration, and attains the lowest $D_\pm$ and the highest within-game correlation of all estimators in each of the nine training runs (Figure 3b). MC and TMC at the same budget remain 1.7 to 1.9 times further from the exact value, 1.5 times the queries still leaves them 24% to 39% behind, and the samplers of AFedSV and FedDSV, the latter spending 510 queries, are 1.6 to 1.8 times further away. Paired bootstrap intervals exclude zero for every comparison, and ten sampler seeds move FedACE by at most 0.005 (Appendix B.3). The anchor value, the only quantity the ledger books, also has the lowest absolute error on every dataset (Table 7). Most arrivals need no estimation at all, since the route answers

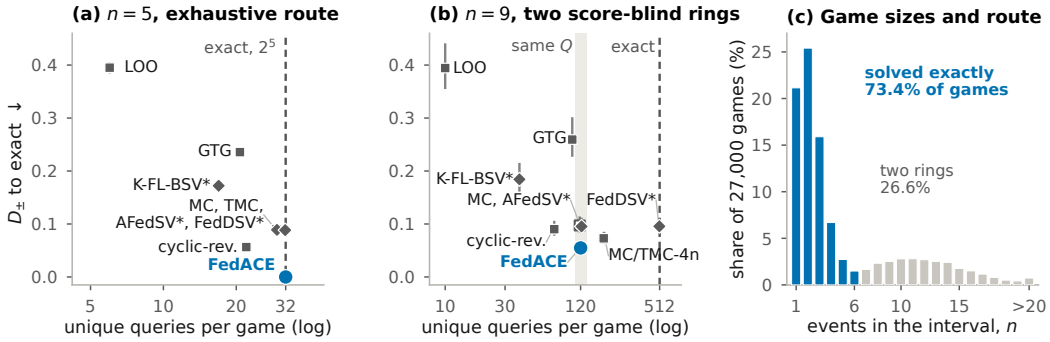

Figure 3: **FedACE is exact when affordable and closest otherwise.** (a) At $n = 5$ the exhaustive route is exact. (b) At $n = 9$ FedACE beats every estimator at the same budget (shaded) or above, with bars spanning dataset means and diamonds marking asynchronous samplers. (c) Games with $n \leq 6$, 73.4% of all, are solved exactly.

Table 2: Guarantees of Section 3 recomputed from the recorded artefacts at each of the 27,000 arrivals. Appendix B.2 lists every check.

Guarantee	Check at every arrival	Result
Reconstruction (Prop. 2)	composite of C_e against the applied parameters	27,000/27,000
Efficiency (Cor. 1)	$\left\lvert\sum_i \phi_i^e - (U_e(\theta_{e+1}) - U_e(\theta_{d_e}))\right\rvert$	$\leq 4.4 \times 10^{-16}$
Isolation (Def. 3)	parameters, buffers and random states restored	27,000/27,000
Single entry (Prop. 6)	ledger entries credited by one arrival	1
Routing (Prop. 5)	exact branch iff $2^n \leq B$, queries within bound	0 exceptions
No look-ahead (Def. 1)	valuation reads a later arrival	never

Table 3: Replacing one component at a time on all 20,488 complete tables. The last column is the client-ranking Kendall correlation with the exact ranking, mean over nine runs (lowest run).

Module	Variant	changed	sign flips	$\lvert\Delta\phi\rvert$	Kendall↑
	FedACE vs exact	681	2	.0012	0.997 (0.992)
Interval	last five events only	1,083	7	.0041	0.987 (0.978)
	newest event only	14,779	1,783	.0469	0.918 (0.891)
Solver	one random ring	681	1	.0017	0.997 (0.992)
	fixed ring over arrival order	681	1	.0016	0.997 (0.990)
	no exhaustive branch	3,626	32	.0035	0.994 (0.991)
Ledger	credit every event of every game	all	n/a	n/a	0.787 (0.709)

73.4% of the 27,000 games exactly (Figure 3c). Enumerating every game would cost 2.4×10^{10} queries, whereas FedACE spends 2.2×10^{6}, four orders of magnitude fewer.

4.3 Every guarantee holds at every arrival

FedACE is auditable by construction, and every guarantee holds at every one of the 27,000 arrivals (Table 2). The replay reconstructs the applied parameters at each arrival, efficiency holds to within 4.4×10^{-16} and exactly in 19,778 games, and the live parameters, buffers and random states are restored after every query. The router takes the exact branch exactly when $2^n \leq B$, no game exceeds the query bound of Proposition 5, and no valuation reads a later arrival, so on the recorded event sequence training with FedACE is bit identical to training without it.

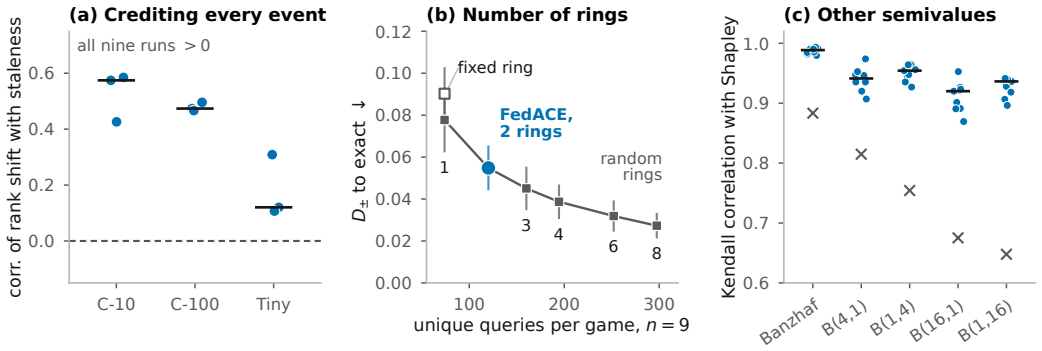

Figure 4: **Each component matters, and client rankings survive the declared solution concept.** (a) Crediting every event shifts the ranking toward slow clients in all nine runs. (b) Random rings beat a fixed ring at equal cost, and error falls as rings are added. (c) Other semivalues reorder events within a game (crosses, $n = 9$) but barely move client rankings (dots, one per run).

4.4 What each component contributes

Every module matters (Table 3). Crediting every event of every game instead of only the anchor counts each update 4.99 times on average and up to 34 times, drops the Kendall correlation of the client ranking to 0.787, and shifts it toward slow clients in all nine runs (Figure 4a). Keeping only the newest event of the dependency interval reverses the sign of the credited value in $1{,}783$ of the $14{,}779$ games it changes, and reversing the composition order changes the event ranking in all 24 games of Figure 1b. Size routing moves the client ranking by at most 0.008 in Kendall correlation, while two random rings cut the gap of one ring by 30% and of a fixed ring by 39% (Figure 4b), and enumerating up to six events raises exact coverage from 62.5% to 73.4% for 0.6% more queries.

4.5 Robustness and positioning

Client rankings are stable under the declared choices. Exact Banzhaf and Beta semivalues (Kwon & Zou, 2022) reorder events within a game, with the Kendall correlation to the Shapley value at $n = 9$ falling to between 0.648 and 0.883. Averaging anchor values over the arrivals of each client absorbs almost all of this change, so the client ranking keeps a mean Kendall correlation between 0.911 and 0.987 and a Spearman correlation above 0.97 in every run (Figure 4c). Replacing its public set by a disjoint held-out set keeps a within-game Spearman correlation of 0.89 between exact values on three development runs, and FedACE stays within $D_\pm = 0.036$ of the exact value under its declared utility (Appendix B.5). Replayed as passive observers on the same records, the published evaluators agree with the exact value of the declared game at most 0.442, for CGSV, which is also the only one positive in all nine runs. K-FL-BSV and FedMS are negatively correlated on average, ShapFed, SPACE and TMC-Shapley turn negative when their per-arrival score is replaced by the cumulative total, and for seven of the eight the sign of the agreement flips between datasets (Appendix B.7). Their scores are therefore not interchangeable, whereas a declared estimand with an exact reference makes any two evaluators comparable.

5 Conclusion

Fully asynchronous FL removes the round on which contribution evaluation was built, and without it published evaluators no longer agree on what they measure. FedACE restores a well-defined target by reading the players, reference state, and composition order of each update's game from the execution record. We proved that this game is well posed, that the record identifies its composition order, and that it coincides with the classical round game under a synchronous batch rule. Its four modules solve 73.4% of the games exactly and, on all audited nine-event games, more accurately per query than every compared estimator in every training run, credit each update exactly once.

AI USE STATEMENT

We used large language model assistants in this work. They helped polish and refine the text of the paper, and check the consistency of notation and proofs. They did not generate experimental data. The authors reviewed every AI-assisted passage, script and derivation, reran the analysis scripts against the recorded logs, and take full responsibility for its content, including text, claims and artifacts.

ETHICS STATEMENT

This work uses public image benchmarks and involves no human subjects. Contribution scores can affect compensation and participation, and FedACE makes the valued quantity explicit, auditable and free of feedback into training, so that such decisions rest on a declared and checkable basis.

REPRODUCIBILITY STATEMENT

Section 4.1 and Appendix B.1 specify datasets, partitions, model, training schedule, staleness rule and seeds. Section 3.5 fixes the solver configuration, and Appendix A proves every statement of Section 3. Table 5 lists the checks recomputed at every arrival. Code and traces will be released.

REFERENCES

Javier Castro, Daniel Gómez, and Juan Tejada. Polynomial calculation of the shapley value based on sampling. *Computers & Operations Research*, 36(5):1726–1730, 2009. ISSN 0305-0548. doi: https://doi.org/10.1016/j.cor.2008.04.004. URL https://www.sciencedirect.com/science/article/pii/S0305054808000804. Selected papers presented at the Tenth International Symposium on Locational Decisions (ISOLDE X).

Yi-Chung Chen, Hsi-Wen Chen, Shun-Gui Wang, and Ming-syan Chen. Space: Single-round participant amalgamation for contribution evaluation in federated learning. In A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 6422–6441. Curran Associates, Inc., 2023. doi: 10.52202/075280-0281. URL https://proceedings.neurips.cc/paper_files/paper/2023/file/14a812fa4b6bf244d055e37a7cd2f557-Paper-Conference.pdf.

Zhenan Fan, Huang Fang, Xinglu Wang, Zirui Zhou, Jian Pei, Michael Friedlander, and Yong Zhang. Fair and efficient contribution valuation for vertical federated learning. In B. Kim, Y. Yue, S. Chaudhuri, K. Fragkiadaki, M. Khan, and Y. Sun (eds.), *International Conference on Learning Representations*, volume 2024, pp. 14553–14572, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/3eaca8256ac27d5e67d604e0f00401f4-Paper-Conference.pdf.

Amirata Ghorbani and James Zou. Data shapley: Equitable valuation of data for machine learning. In Kamalika Chaudhuri and Ruslan Salakhutdinov (eds.), *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 2242–2251. PMLR, 09–15 Jun 2019. URL https://proceedings.mlr.press/v97/ghorbani19c.html.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

Yuhui Jiang, Xingjian Lu, Wei Mao, and Ying Lin. Integrating staleness and shapley value consistency for efficient k-asynchronous federated learning. In *2023 IEEE International Conference on Big Data (BigData)*, pp. 680–689. IEEE, 2023.

Peter Kairouz and H. Brendan McMahan. Advances and open problems in federated learning. *Foundations and Trends in Machine Learning*, 14(1-2):1–210, 06 2021. ISSN 1935-8237. doi: 10.1561/2200000083. URL https://doi.org/10.1561/2200000083.

Anastasiia Koloskova, Sebastian U Stich, and Martin Jaggi. Sharper convergence guarantees for asynchronous sgd for distributed and federated learning. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural Information Processing Systems*, volume 35, pp. 17202–17215. Curran Associates, Inc., 2022. doi: 10.52202/068431-1251. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/6db3ea527f53682657b3d6b02a841340-Paper-Conference.pdf.

Alex Krizhevsky, Geoffrey Hinton, et al. Learning multiple layers of features from tiny images.(2009), 2009.

Yongchan Kwon and James Zou. Beta shapley: a unified and noise-reduced data valuation framework for machine learning. In *Proceedings of the 25th International Conference on Artificial Intelligence and Statistics (AISTATS)*, volume 151 of *Proceedings of Machine Learning Research*, pp. 8780–8802, 2022.

Yann Le, Xuan Yang, et al. Tiny imagenet visual recognition challenge. *CS 231N*, 7(7):3, 2015.

Kaijia Lei, Xuebin Ren, Shusen Yang, Xiaocheng Wang, and Fangyuan Zhao. Feddsv: Shapley value-based contribution estimation in federated learning with dynamic participation. *IEEE Transactions on Mobile Computing*, 24(10):11019–11033, 2025. doi: 10.1109/TMC.2025.3574784.

Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. Federated optimization in heterogeneous networks. In I. Dhillon, D. Papailiopoulos, and V. Sze (eds.), *Proceedings of Machine Learning and Systems*, volume 2, pp. 429–450, 2020. URL https://proceedings.mlsys.org/paper_files/paper/2020/file/1f5fe83998a09396ebe6477d9475ba0c-Paper.pdf.

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.

Zelei Liu, Yuanyuan Chen, Han Yu, Yang Liu, and Lizhen Cui. Gtg-shapley: Efficient and accurate participant contribution evaluation in federated learning. *ACM Transactions on intelligent Systems and Technology (TIST)*, 13(4):1–21, 2022.

Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*, pp. 1273–1282. Pmlr, 2017.

Rory Mitchell, Joshua Cooper, Eibe Frank, and Geoffrey Holmes. Sampling permutations for shapley value estimation. *Journal of Machine Learning Research*, 23(43):1–46, 2022.

John Nguyen, Kshitiz Malik, Hongyuan Zhan, Ashkan Yousefpour, Mike Rabbat, Mani Malek, and Dzmitry Huba. Federated learning with buffered asynchronous aggregation. In Gustau Camps-Valls, Francisco J. R. Ruiz, and Isabel Valera (eds.), *Proceedings of The 25th International Conference on Artificial Intelligence and Statistics*, volume 151 of *Proceedings of Machine Learning Research*, pp. 3581–3607. PMLR, 28–30 Mar 2022. URL https://proceedings.mlr.press/v151/nguyen22b.html.

Lloyd S. Shapley. A value for n-person games. In Harold W. Kuhn and Albert W. Tucker (eds.), *Contributions to the Theory of Games, Volume II*, number 28 in Annals of Mathematics Studies, pp. 307–317. Princeton University Press, 1953.

Nurbek Tastan, Samar Fares, Toluwani Aremu, Samuel Horvath, and Karthik Nandakumar. Redefining contributions: Shapley-driven federated learning, 2024. URL https://arxiv.org/abs/2406.00569.

Zhaoxuan Wu, Mohammad Mohammadi Amiri, Ramesh Raskar, and Bryan Kian Hsiang Low. Incentive-aware federated learning with training-time model rewards. In B. Kim, Y. Yue, S. Chaudhuri, K. Fragkiadaki, M. Khan, and Y. Sun (eds.), *International Conference on Learning Representations*, volume 2024, pp. 38263–38298,

2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/file/a6250ffb9403aa7fa5304f525b5ae0e4-Paper-Conference.pdf.

Cong Xie, Sanmi Koyejo, and Indranil Gupta. Asynchronous federated optimization, 2020. URL https://arxiv.org/abs/1903.03934.

Haotian Xu, Zhaorui Zhang, Sheng Di, Benben Liu, Khalid Ayed Alharthi, and Jiannong Cao. Fedfa: A fully asynchronous training paradigm for federated learning. In *Proceedings of the 33rd International Joint Conference on Artificial Intelligence (IJCAI)*, pp. 5281–5288, 2024.

Xinyi Xu, Lingjuan Lyu, Xingjun Ma, Chenglin Miao, Chuan Sheng Foo, and Bryan Kian Hsiang Low. Gradient driven rewards to guarantee fairness in collaborative machine learning. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems*, volume 34, pp. 16104–16117. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper_files/paper/2021/file/8682cc30db9c025ecd3fee433f8ab54c-Paper.pdf.

Chengxu Yang, Qipeng Wang, Mengwei Xu, Zhenpeng Chen, Kaigui Bian, Yunxin Liu, and Xuanzhe Liu. Characterizing impacts of heterogeneity in federated learning upon large-scale smartphone data. In *Proceedings of the Web Conference 2021*, WWW '21, pp. 935–946, New York, NY, USA, 2021. Association for Computing Machinery. ISBN 9781450383127. doi: 10.1145/3442381.3449851. URL https://doi.org/10.1145/3442381.3449851.

Mengwei Yang, Baturalp Buyukates, and Athina Markopoulou. Rewarding the rare: Maverick-aware shapley valuation in federated learning. *Transactions on Machine Learning Research*, 2025. ISSN 2835-8856. URL https://openreview.net/forum?id=JtybGfTUdq.

Liyong Zhang, Guangyong Shang, Rui Jin, Yu Li, Long Ma, and Feng Dong. Shapley value-based fair asynchronous federated learning. *Computer Communications*, 257:108618, 2026. ISSN 0140-3664. doi: https://doi.org/10.1016/j.comcom.2026.108618. URL https://www.sciencedirect.com/science/article/pii/S0140366426002082.

A Deferred theory and proofs

Throughout we use the notation of Section 3. States are indexed by version, so θ_t is the global model after t updates and $\theta_{e+1} = T_e(\theta_e)$ with $T_j(\theta) = (1-\alpha_j)\theta + \alpha_j \ell_j$. The dependency interval is $C_e = \{d_e, \dots, e\}$ of size $n_e = \tau_e + 1$, the order $\prec$ is the factual arrival order, ϕ_e^e is the anchor value of event e, and Φ_c is the client ledger of Eq. 6.

A.1 Proof of Theorem 1

Lemma 1 (Well-posedness). *$\bar{v}_e$ is a function $2^{C_e} \to \mathbb{R}$ with $\bar{v}_e(\varnothing) = 0$.*

Proof. $\prec$ is a total order on the finite set C_e, so every $S \subseteq C_e$ has exactly one $\prec$-increasing enumeration and Eq. 2 depends on S alone. At $S = \varnothing$ the composite is θ_{d_e}, so $\bar{v}_e(\varnothing) = 0$. □

A set function need not be invariant to the order of its inputs. It must assign one value to each subset, and the fixed order $\prec$ supplies exactly that, so non-commutativity is no obstruction.

Statement. $(C_e, \bar{v}_e)$ is a finite cooperative game, so a unique value on C_e satisfies efficiency, symmetry, the null player property and linearity, and it is given by Eq. 3.

Proof. Shapley's theorem requires a finite player set, a real valued function on its subsets, and the normalisation $v(\varnothing) = 0$. The set C_e is finite by Definition 1. Lemma 1 shows that $\bar{v}_e$ is a function on 2^{C_e} and that $\bar{v}_e(\varnothing) = 0$. The three requirements hold, so the classical uniqueness theorem applies to $(C_e, \bar{v}_e)$ without modification, and the unique value satisfying efficiency, symmetry, the null player property and linearity is given by Eq. 3. No axiom is weakened or added. □

A.2 Proof of Proposition 1

Statement. Let R be the set of clients a round dispatches to, identify each client with its update, and let e be the last arrival of the round. Suppose every $i \in R$ computes ℓ_i from the same dispatched state θ_t and the server aggregates any $S \subseteq R$ by

$$A_S(\theta) \;=\; \Big(1 - \sum_{i \in S} \omega_i\Big)\theta \;+\; \sum_{i \in S} \omega_i\, \ell_i, \qquad \omega_i \geq 0, \qquad \sum_{i \in R} \omega_i = 1, \tag{7}$$

where ω_i is the aggregation weight of client i. At $S = R$ this is the classical average, at $S = \varnothing$ it is the identity, and at every S it depends on S as a set. Then $C_e = R$, the composite $\Theta_e(S) = A_S(\theta_t)$ does not depend on $\prec$, and $\bar{v}_e(S) = U_e(A_S(\theta_t)) - U_e(\theta_t)$ is the classical synchronous characteristic function, so ϕ^e is the classical Shapley value of the round game. The statement concerns a second composition protocol. Definition 2 composes the sequential operators of Eq. 1, which differ from A_S even when all updates share one reference state, and the proposition shows that composing coalitions with the round's batch rule instead recovers the classical game.

Proof. Let t be the number of updates the server had applied when the round was dispatched, and let R be the set of clients it dispatched to. By hypothesis every $i \in R$ computes ℓ_i from θ_t, so every member of R carries the same dispatch version t.

Player set. Definition 1 sets $C_e = \{d_e, \ldots, e\}$, the events that completed between the dispatch of e and its application. Since the round is applied by one operator, the round contributes $|R|$ arrival events, which occupy consecutive indices ending at the event e that completes the round, and each of them has dispatch version t. Hence $d_e = t$ and $C_e = \{t, \ldots, e\}$ is exactly the set of updates the round collected, of size $n_e = |R|$. Each client of R contributes one update to the round, so the map $c(\cdot)$ of Section 3.2 restricts to a bijection between C_e and R and we identify the two. The synchronous case is precisely the one in which players and clients coincide, which is why the distinction does not appear in the classical formulation.

Composition. Definition 2 composes the members of S in the unique $\prec$-increasing order. Under Eq. 7 the server does not compose members one at a time. It applies the single operator A_S, whose value depends on S through the multiset $\{(\omega_i, \ell_i)\}_{i \in S}$ and not through any enumeration of it. Hence for every total order σ on C_e we have $\Theta_e^\sigma(S) = A_S(\theta_t)$, and the dependence on $\prec$ in Definition 2 is vacuous.

Characteristic function. Definition 2 requires $\Theta_e(\varnothing) = \theta_t$, which Eq. 7 supplies because $A_\varnothing$ is the identity. Substituting gives $\bar{v}_e(S) = U_e\big(A_S(\theta_t)\big) - U_e(\theta_t)$, the characteristic function of the classical round game with player set R and utility U_e.

Value. Theorem 1 applies to $(C_e, \bar{v}_e)$ as before, and since $(C_e, \bar{v}_e)$ now coincides with the classical round game, the unique value satisfying the four axioms coincides with the classical Shapley value of that game. The factual-prefix reconstruction of Proposition 2 holds automatically, because $A_R(\theta_t)$ is by construction the state the server reaches at the end of the round, and there is a single round aggregate to check. □

Remark 1. Both hypotheses are needed and each fails separately under asynchrony. Distinct dispatch versions inside one dependency interval make C_e depend on the arrival rather than on a round, and sequential contraction makes $\Theta_e(S)$ depend on an ordering. Proposition 1 therefore identifies exactly which two properties synchrony supplies for free, and Section 3.2 recovers them without assuming either.

A.3 Proof of Proposition 2

Statement. For the factual arrival order $\prec$, every prefix composite satisfies Eq. 4. In particular, the grand coalition reconstructs the state reached by the recorded execution.

Proof. The server applies T_j for $j = d_e, \ldots, e$ in increasing j, so by Eq. 1 and induction on k the state after the arrivals $d_e, \ldots, k$ is θ_{k+1}. The set $\{d_e, \ldots, k\}$ is a $\prec$-prefix of C_e, and its composite under Definition 2 is exactly that recursion started at θ_{d_e}. Hence the factual order satisfies Eq. 4 for every k, including $k = e$, where $\Theta_e(C_e) = \theta_{e+1}$.

This proves reconstruction for the declared factual order. Whether the prefix endpoints also identify the order is the content of Proposition 3, which needs a non-degeneracy condition that the present statement does not. □

Remark 2 (Verification)**.** Eq. 4 is a property of the implementation as well as of the construction, and it can fail through any mismatch between the live update and the replay composer, for example an unrestored normalisation buffer. We therefore verify it event by event rather than assume it, by comparing the parameters of the grand coalition composite with the applied task parameters at every arrival. The claim is about parameters. Normalisation buffers are recalibrated per coalition by Definition 3 and are separately checked for restoration, not for reconstruction. Table 5 reports the outcome together with the other properties of this section that are checked the same way, none of which has an exception over the 27,000 arrivals.

A.4 Proof of Proposition 3

Statement. For $d_e \le j < k \le e$ let $\gamma_{jk} = \prod_{m=j}^{k-1}(1-\alpha_m)$ and

$$\bar{\ell}_{jk} \;=\; \frac{1}{1-\gamma_{jk}} \sum_{i=j}^{k-1} \alpha_i \Big(\prod_{m=i+1}^{k-1} (1-\alpha_m) \Big) \ell_i, \tag{8}$$

the effective aggregate of the updates $j, \dots, k-1$ that k would overtake, which is a convex combination of $\ell_j, \dots, \ell_{k-1}$. If every $\alpha_j > 0$ and $\ell_k \neq \bar{\ell}_{jk}$ for all $d_e \le j < k \le e$, then no total order on C_e other than $\prec$ satisfies Eq. 4 when used in place of $\prec$ in Eq. 2. For fixed coefficients and $\ell_j \in \mathbb{R}^p$ the excluded configurations form a finite union of affine subspaces of codimension p.

Proof. Write $P_k = \{d_e, \dots, k\}$ and let σ be a total order on C_e that satisfies Eq. 4. Suppose $\sigma \neq \prec$. Then there is a smallest k such that the restriction of σ to P_k differs from the factual order on P_k. Necessarily $k > d_e$, the restriction to P_{k-1} is factual, and the restriction to P_k inserts k immediately before some j with $d_e \le j \le k-1$, leaving $j, \dots, k-1$ in factual order after it.

Let $y = T_{j-1} \circ \cdots \circ T_{d_e}(\theta_{d_e})$, with $y = \theta_{d_e}$ when $j = d_e$, and let $F = T_{k-1} \circ \cdots \circ T_j$. The composite of P_k under σ is $F(T_k(y))$, while Eq. 4 requires it to equal $\theta_{k+1} = T_k(F(y))$. Every T_m is affine with slope $1-\alpha_m$, so $F(x) = \gamma_{jk}\, x + b$ with $b = \sum_{i=j}^{k-1} \alpha_i \big(\prod_{m=i+1}^{k-1}(1-\alpha_m)\big)\ell_i$. Substituting gives

$$T_k(F(y)) - F(T_k(y)) \;=\; \alpha_k \big[(1-\gamma_{jk})\, \ell_k - b \big],$$

which does not depend on y. The coefficients of b are nonnegative and sum to $1-\gamma_{jk}$ by telescoping, and $1-\gamma_{jk} > 0$ because every $\alpha_m > 0$. Since $\alpha_k > 0$, the requirement at k holds if and only if $\ell_k = b/(1-\gamma_{jk}) = \bar{\ell}_{jk}$, which the hypothesis excludes. Hence $\sigma = \prec$. □

Remark 3 (The condition is sharp)**.** Take three updates with $\alpha_1 = \alpha_2 = \alpha_3 = \frac{1}{2}$, scalar models $\ell = (2, 8, 6)$ and reference 0. Then $\bar{\ell}_{13} = \frac{1}{3}\cdot 2 + \frac{2}{3}\cdot 8 = 6 = \ell_3$. The order $(3, 1, 2)$ gives the prefix endpoints 1, 4.5 and 5.25, the factual ones, yet on the coalition $\{1, 3\}$ it gives $T_1(T_3(0)) = 2.5$ against $T_3(T_1(0)) = 3.5$ for the factual order. Perturbing ℓ_3 removes the second order.

A.5 Proof of Corollary 1

Statement. $\sum_{i \in C_e} \phi_i^e = U_e(\theta_{e+1}) - U_e(\theta_{d_e})$.

Proof. Efficiency gives $\sum_{i \in C_e} \phi_i^e = \bar{v}_e(C_e)$. By Definition 2, $\bar{v}_e(C_e) = U_e(\Theta_e(C_e)) - U_e(\theta_{d_e})$, and Eq. 4 at $k = e$ gives $\Theta_e(C_e) = \theta_{e+1}$. Substituting gives $\sum_{i \in C_e} \phi_i^e = U_e(\theta_{e+1}) - U_e(\theta_{d_e})$, which is the utility change the deployment underwent across the dependency interval of event e as measured by the declared utility. □

A.6 Proof of Proposition 4

Statement. Under Assumption 2 the marginal $\bar{v}_e(S \cup \{i\}) - \bar{v}_e(S)$ is the utility difference between the states that the server's own rule produces from $S \cup \{i\}$ and from S under the frozen-record protocol. If instead α depends on a contribution score, the recorded coefficients alone do not in general recover that difference.

Proof. Assume Assumption 2, so $\alpha_j = \lambda\, w(\tau_j)$ with τ_j determined by the dispatch record of event j alone. Fix S and $i \notin S$. The staleness τ_j of any $j \in S \cup \{i\}$ is computed from the recorded dispatch and completion indices of j, which the frozen-record protocol keeps whichever other updates are retained. Hence the coefficient the rule assigns to each j is the same whether or not i is present, and $\Theta_e(S \cup \{i\})$ and $\Theta_e(S)$ are built from the coefficients the rule would itself have used in each case. The difference $\bar{v}_e(S \cup \{i\}) - \bar{v}_e(S)$ is therefore the utility difference the server's own rule would have produced had i been absent, with local computation held at its recorded values.

Now drop Assumption 2 and let $\alpha_j = g(\psi)$ for a rule g and a score vector ψ computed from the run. The recorded coefficients are $g(\psi)$ with ψ evaluated on a trajectory in which i participated. Had i been absent, the score vector would have been some ψ' and the rule would have set $g(\psi')$, so the recorded coefficients are not the ones the rule would have used. Recovering ψ' in general requires the trajectory that $g(\psi')$ would have produced, which the recorded coefficients do not contain, so they do not in general recover the difference. □

A.7 What the signal measures under withheld effort

Corollary 2 (Withholding effort is not being a null player)**.** *If client $c(e)$ returns the model it received without training, so $\ell_e = \theta_{d_e}$, then $T_e(\theta) - \theta = \alpha_e(\theta_{d_e} - \theta)$, which is nonzero whenever $\alpha_e > 0$ and $\theta \neq \theta_{d_e}$. Such an event is in general not a null player, and ϕ_e^e is in general nonzero.*

FedACE therefore values what an update did to the global model rather than the effort behind it.

Proof. With $\ell_e = \theta_{d_e}$ the operator is $T_e(\theta) = (1 - \alpha_e)\theta + \alpha_e\theta_{d_e}$, so $T_e(\theta) - \theta = \alpha_e(\theta_{d_e} - \theta)$. If $\alpha_e > 0$ and $\theta \neq \theta_{d_e}$ then $T_e \neq \mathrm{id}$. Event e is a null player of $(C_e, \bar{v}_e)$ only if $U_e(T_e(\Theta_e(S))) = U_e(\Theta_e(S))$ for every $S \subseteq C_e \setminus \{e\}$, which fails as soon as U_e separates $\Theta_e(S)$ from its displacement by $\alpha_e(\theta_{d_e} - \Theta_e(S))$ for one such S. Hence e is in general not a null player and $\phi_e^e \neq 0$.

Two limits are worth recording. When $\tau_e = 0$ we have $d_e = e$, so $C_e = \{e\}$ and the only subset the game evaluates is $\varnothing$, whose composite is θ_{d_e}. On that state $T_e(\theta_{d_e}) = (1 - \alpha_e)\theta_{d_e} + \alpha_e\theta_{d_e} = \theta_{d_e}$, so T_e acts as the identity on every state the game evaluates, the single marginal of Eq. 3 vanishes and $\phi_e^e = 0$ identically. This holds for every choice of utility, since no property of U_e is used. It is the one point at which the construction is obliged to report exactly zero. When $\tau_e > 0$ the displacement is nonzero and its sign is governed by the curvature of U_e along $\theta_{d_e} - \Theta_e(S)$ rather than by the direction of the displacement alone. □

A.8 Proof of Proposition 5

Statement. Write $n = n_e$ for the size of the game, let B be the enumeration budget and let r be the number of rings. Exact evaluation uses 2^n distinct utility queries, the ring estimator of Eq. 5 uses at most $2 + r\, n(n-1)$, it equals Eq. 3 for $n \leq 3$, and the routing rule evaluates exactly whenever $2^n \leq B$ and reads only n.

Proof. Eq. 3 sums over all $S \subseteq C_e \setminus \{i\}$ for every i, and the union of $\{S, S \cup \{i\}\}$ over these pairs is 2^{C_e}, so exact evaluation needs the value of every one of the 2^n subsets and no fewer.

For the ring estimator, each path is a permutation of C_e and contributes the $n + 1$ prefixes of that permutation. The empty prefix and the full prefix are shared by every path, which accounts for two distinct masks. Fix one ring π, read as a cyclic order on C_e. The length k prefix of the rotation of π beginning at position i is the set of k consecutive entries of π starting at i, so the prefixes of the n rotations are exactly the contiguous arcs of π. The length k prefix of a reversed rotation is the set of k consecutive entries ending at a given position, which is again a contiguous arc of π, so reversals contribute no further masks. There are n arcs of each length k with $1 \leq k \leq n - 1$, hence one ring contributes at most $n(n-1)$ proper nonempty masks and r rings at most $r\, n(n-1)$. Adding the empty and full masks gives $2 + r\, n(n-1)$. The bound is stated per ring because the rings are drawn independently, so arcs of distinct rings need not coincide.

For $n \leq 3$ the n rotations of any ring and their reversals are $2n$ paths that contain every one of the $n!$ permutations of C_e exactly $2n/n!$ times, which is twice for $n \in \{1, 2\}$ and once for $n = 3$. Eq. 5 then averages the marginal uniformly over all permutations, which is Eq. 3.

Table 4: Recorded traces and the event games they induce. Each dataset contributes three independent training runs of 3,000 server updates over 50 clients, and every update defines one game. Accuracy is the final test accuracy of the global model in percent, mean and standard deviation over the three runs, averaged over the client test splits. A stale arrival is applied after at least one intervening update, and n counts the events in the dependency interval. Exhaustive is the share of games answered by enumeration. Audited counts the games with $n = 9$, whose complete coalition table is computed independently of the route. Together they give exact references on 20,488 games.

Dataset	Classes	Accuracy	Games	Stale	Mean n	Max n	Exhaustive	Audited
CIFAR-10	10	48.4 ± 2.3	9,000	77.2%	4.99	34	74.4%	201
CIFAR-100	100	29.9 ± 1.1	9,000	78.8%	4.99	27	72.8%	221
Tiny-ImageNet	200	14.4 ± 0.8	9,000	80.6%	4.99	25	72.9%	259
All			27,000	78.9%	4.99	34	73.4%	681

The routing rule compares 2^n with the budget B and reads nothing else. In particular it does not evaluate U_e, does not consult a reference value, and does not read any label attached to a client, so the branch taken at event e is a function of n_e alone. □

A.9 Proof of Proposition 6

Statement. Each arrival event is charged once, as the anchor of its own game, and contributes to no other entry of Eq. 6, so Φ_c is well defined although dependency intervals overlap.

Proof. Each arrival event e induces exactly one game $(C_e, \bar{v}_e)$, in which e occupies the last slot of $\prec$. Eq. 6 adds to client $c(e)$ the anchor value ϕ_e^e of that game and nothing else, so event e contributes to exactly one entry, namely the entry of its own client. An event $j < e$ belongs to C_e and receives a value ϕ_j^e inside that game, but ϕ_j^e enters no entry. It is a contextual quantity used to define the anchor's marginal. Consequently, although the dependency intervals of distinct arrivals overlap, no event is charged twice, and Φ_c is the sum over the events of client c of their own anchor values. Note that $c(\cdot)$ is many to one, so a client may hold several slots of one C_e. This changes nothing in the argument, because the accounting is indexed by events and each event has one anchor. □

Remark 4. This is also why efficiency is a per game statement. Summing ϕ_e^e across events aggregates anchors drawn from different games with different reference states, and the resulting total is not claimed to equal any single utility difference. The artefacts record this explicitly.

B Additional experimental results

B.1 Recorded traces

Table 4 summarises the nine recorded runs. CIFAR-10 uses seeds 3 to 5, which were not used during development, and CIFAR-100 and Tiny-ImageNet use seeds 0 to 2. The staleness discount is the FedAsync hinge with $a = 1$ and $b = 4$. Batch-normalisation statistics are recalibrated on the public inputs for each coalition. Fidelity is measured on every game with $n = 5$ and every game with $n = 9$, all of which have complete coalition tables.

B.2 Protocol certificate

B.3 Fidelity details

Table 6 adds TMC, the same-budget TMC, the ablation and top-1 agreement to Table 1, and Table 7 reports per-event diagnostics at $n = 9$. None of the 1,417 games has a degenerate exact vector or tied exact values, and the Spearman correlation is defined for every estimate. The dagger in Table 1 compares run means, and a paired bootstrap over the nine runs gives an interval for the difference to FedACE that excludes zero for every estimator, for example $[0.040, 0.046]$ against MC at the same budget and $[0.015, 0.019]$ against MC-$4n$. FedACE draws its rings from one fixed sampler seed.

Table 5: Protocol certificate over the 27,000 arrival events of the nine runs of Section 4. Every row is recomputed from the recorded artefacts at every arrival, and no row has an exception.

Statement	Quantity recomputed at every arrival	Result
The valuation refers to the run that actually happened		
Eq. (4)	hash of $\Theta_e(C_e)$ parameters against the applied parameters	27,000/27,000
Def. 2	hash of $\Theta_e(\varnothing)$ against θ_{d_e}	27,000/27,000
Cor. 1	$\left\lvert \sum_i \phi_i^e - (U_e(\theta_{e+1}) - U_e(\theta_{d_e})) \right\rvert$	$\leq 4.4 \times 10^{-16}$
Cor. 1	the same residual, exactly zero	19,778/27,000
The valuation does not disturb the deployment		
Def. 3	live parameters and RNG restored afterwards	27,000/27,000
Def. 3	normalisation buffers restored afterwards	27,000/27,000
Prop. 6	ledger entries credited by one arrival	1 at every arrival
Nothing in the procedure is chosen after seeing an outcome		
Prop. 5	exact branch taken exactly when $2^n \leq B$	0 exceptions
Prop. 5	distinct queries against $2 + r\,n(n-1)$	0 exceptions
Sec. 3.5	ring reads a utility or an oracle	never
Def. 1	valuation reads a later arrival	never
Def. 1	dependency interval truncated	never
Sec. 3.3	estimand, utility and composition order	single valued

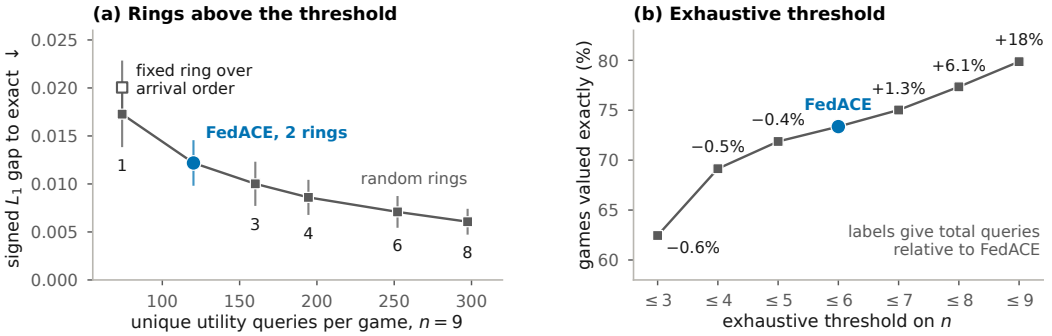

Figure 5: Solver ablation. (a) Signed L_1 gap to the exact value against unique queries on all 681 audited games with $n = 9$, for one to eight random rings and for a fixed ring over arrival order, with the standard deviation over nine runs. (b) Share of the 27,000 games valued exactly as the exhaustive threshold moves, with total queries relative to FedACE above each point.

Over ten sampler seeds its $D_{\pm}$ at $n = 9$ has a standard deviation between 0.0010 and 0.0015 per dataset, and the worst seed still lies below MC-$4n$ on every dataset. Figure 5 resolves the solver rows of Table 3 by the number of rings and by the exhaustive threshold. The FedDSV schedule keeps its default residual threshold, which on the declared game selects nearly every coalition size, so its threshold phase queries almost all coalitions before it samples.

B.4 Solution concept

Figure 6 resolves the solution-concept rows of Table 8 by the size of the dependency interval.

B.5 Utility

The utility is declared, and part of that declaration is the public set on which it is computed. Table 9 replaces this set by a disjoint held-out set of the same size on three CIFAR-10 development runs. The exact values move by $D_{\pm} = 0.146$, whereas FedACE stays at 0.036 from the exact value of the quantity it declares. FedACE therefore tracks its declared utility closely, and the larger distance to the held-out values measures a change of estimand rather than an estimation error.

Table 6: Full version of Table 1, adding TMC, the same-budget TMC and the ablation FedACE$^-$, which removes the exhaustive branch and runs the two rings at every size and therefore coincides with FedACE at $n = 9$. t_1 is the probability that the event an estimator ranks first is an event the exact value ranks first. The exact reference has $D_\pm = 0$ and $t_1 = 1$ by definition. GTG-Shapley is reconstructed on the same table with its guided start taken from the singleton marginal.

		$D_\pm$ to exact ↓			t_1 ↑		
Method	Q	C-10	C-100	Tiny	C-10	C-100	Tiny
$n = 5$, exhaustive route, 736 games, exact enumeration takes 32 queries							
LOO	6	$.405_{\pm.027}$	$.396_{\pm.015}$	$.384_{\pm.036}$	$.597_{\pm.078}$	$.610_{\pm.019}$	$.632_{\pm.053}$
cyclic-reverse	22	$.064_{\pm.007}$	$.053_{\pm.004}$	$.053_{\pm.004}$	$.913_{\pm.007}$	$.903_{\pm.044}$	$.957_{\pm.003}$
MC-$4n$	29	$.096_{\pm.008}$	$.085_{\pm.005}$	$.087_{\pm.001}$	$.867_{\pm.024}$	$.844_{\pm.006}$	$.881_{\pm.006}$
TMC-$4n$	29	$.097_{\pm.008}$	$.086_{\pm.005}$	$.088_{\pm.001}$	$.867_{\pm.022}$	$.840_{\pm.006}$	$.879_{\pm.006}$
GTG-Shapley	21	$.234_{\pm.020}$	$.239_{\pm.010}$	$.234_{\pm.031}$	$.768_{\pm.038}$	$.694_{\pm.087}$	$.736_{\pm.038}$
K-FL-BSV sampler*	17	$.183_{\pm.014}$	$.165_{\pm.007}$	$.168_{\pm.001}$	$.745_{\pm.011}$	$.720_{\pm.003}$	$.775_{\pm.005}$
AFedSV sampler*	29	$.095_{\pm.009}$	$.085_{\pm.005}$	$.087_{\pm.002}$	$.868_{\pm.024}$	$.845_{\pm.009}$	$.878_{\pm.006}$
FedDSV sampler*	32	$.096_{\pm.009}$	$.084_{\pm.005}$	$.086_{\pm.001}$	$.869_{\pm.022}$	$.848_{\pm.001}$	$.884_{\pm.001}$
FedACE$^-$ *(ablation)*	27	$.044_{\pm.003}$	$.034_{\pm.005}$	$.037_{\pm.006}$	$.905_{\pm.065}$	$.937_{\pm.025}$	$.957_{\pm.022}$
FedACE	32	**0**	**0**	**0**	**1**	**1**	**1**
$n = 9$, two score-blind rings, 681 games, exact enumeration takes 512 queries							
LOO	10	$.441_{\pm.016}$	$.387_{\pm.012}$	$.355_{\pm.008}$	$.598_{\pm.053}$	$.694_{\pm.015}$	$.714_{\pm.038}$
cyclic-reverse	74	$.106_{\pm.001}$	$.087_{\pm.002}$	$.078_{\pm.003}$	$.819_{\pm.114}$	$.828_{\pm.057}$	$.878_{\pm.053}$
MC-$4n$	187	$.084_{\pm.003}$	$.069_{\pm.001}$	$.062_{\pm.002}$	$.853_{\pm.031}$	$.872_{\pm.035}$	$.905_{\pm.013}$
TMC-$4n$	183	$.085_{\pm.004}$	$.070_{\pm.002}$	$.063_{\pm.002}$	$.844_{\pm.026}$	$.867_{\pm.036}$	$.902_{\pm.011}$
MC *(same Q)*	118	$.114_{\pm.004}$	$.094_{\pm.002}$	$.084_{\pm.002}$	$.808_{\pm.032}$	$.825_{\pm.038}$	$.869_{\pm.019}$
TMC *(same Q)*	118	$.113_{\pm.004}$	$.094_{\pm.002}$	$.084_{\pm.002}$	$.807_{\pm.026}$	$.824_{\pm.035}$	$.863_{\pm.018}$
GTG-Shapley	103	$.302_{\pm.011}$	$.250_{\pm.024}$	$.227_{\pm.015}$	$.597_{\pm.016}$	$.475_{\pm.023}$	$.518_{\pm.058}$
K-FL-BSV sampler*	39	$.215_{\pm.005}$	$.176_{\pm.004}$	$.161_{\pm.005}$	$.679_{\pm.033}$	$.703_{\pm.033}$	$.756_{\pm.027}$
AFedSV sampler*	122	$.112_{\pm.004}$	$.092_{\pm.001}$	$.082_{\pm.002}$	$.812_{\pm.028}$	$.826_{\pm.033}$	$.871_{\pm.019}$
FedDSV sampler*	510	$.111_{\pm.003}$	$.093_{\pm.002}$	$.083_{\pm.003}$	$.812_{\pm.028}$	$.824_{\pm.042}$	$.876_{\pm.021}$
FedACE$^-$ *(ablation)*	120	$\mathbf{.068}_{\pm.008}$	$\mathbf{.051}_{\pm.003}$	$\mathbf{.046}_{\pm.003}$	$\mathbf{.910}_{\pm.010}$	$\mathbf{.891}_{\pm.012}$	$\mathbf{.932}_{\pm.017}$
FedACE	120	$\mathbf{.068}^{\dagger}_{\pm.008}$	$\mathbf{.051}^{\dagger}_{\pm.003}$	$\mathbf{.046}^{\dagger}_{\pm.003}$	$\mathbf{.910}^{\dagger}_{\pm.010}$	$\mathbf{.891}_{\pm.012}$	$\mathbf{.932}_{\pm.017}$

Table 7: Per-event diagnostics on all 681 games with $n = 9$, averaged over three seeds per dataset. ρ is the Spearman correlation between estimated and exact event values within a game, sign is the share of events whose value has the exact sign, and AE is the absolute error on the credited event e, the only value that enters the ledger.

	CIFAR-10			CIFAR-100			Tiny-ImageNet		
Method	ρ ↑	sign↑	AE↓	ρ ↑	sign↑	AE↓	ρ ↑	sign↑	AE↓
LOO	.530	.722	.0158	.759	.838	.0035	.809	.861	.0038
cyclic-reverse	.919	.951	.0033	.941	.964	.0008	.952	.962	.0009
MC-$4n$	.948	.966	.0036	.952	**.977**	.0008	.960	**.977**	.0008
TMC-$4n$	.946	.967	.0042	.950	.976	.0012	.959	.976	.0013
MC *(same Q)*	.920	.951	.0049	.929	.968	.0011	.943	.967	.0011
GTG-Shapley	.748	.840	.0105	.858	.907	.0033	.872	.920	.0039
AFedSV sampler*	.923	.953	.0048	.931	.968	.0011	.943	.969	.0011
FedDSV sampler*	.924	.954	.0052	.929	.968	.0017	.945	.969	.0016
FedACE	**.956**	**.970**	**.0024**	**.963**	**.977**	**.0006**	**.973**	.976	**.0007**

B.6 Interval truncation and composition order

Proposition 2 lets us evaluate a truncated interval exactly from the complete table. Composing the $n - k$ oldest events of C_e in factual order from θ_{d_e} reproduces the recorded state θ_{d_e+n-k}, so keeping the last k events is the game $S \mapsto \bar{v}_e(S \cup O)$ with O the older events always present. Table 10 reports the effect on the 20,488 games with complete tables. Keeping one event reduces the value to the one-step gain and flips its sign in 12.1% of the games it changes, and the flip rate falls steadily as more of the interval is kept. Client rankings move less because averaging absorbs

Table 8: Changing the declared solution concept. Alternative semivalues are recomputed exactly from the complete coalition tables, so no estimation error enters. The within-game Kendall correlation compares value vectors with the declared ones on the games with $n = 9$, and the client columns compare rankings by $\bar{\Phi}_c$ with the declared ranking, as a mean over nine runs with the lowest run in parentheses.

	within-game	client ranking	
Solution concept	Kendall	Kendall	Spearman ρ
Solution concept			
Shapley *(declared)*	1.000	1.000	1.000
Banzhaf	0.883	0.987 (0.980)	0.999 (0.998)
Beta$(4, 1)$	0.815	0.940 (0.907)	0.990 (0.984)
Beta$(1, 4)$	0.754	0.951 (0.927)	0.993 (0.985)
Beta$(16, 1)$	0.675	0.911 (0.869)	0.981 (0.971)
Beta$(1, 16)$	0.648	0.927 (0.896)	0.987 (0.974)

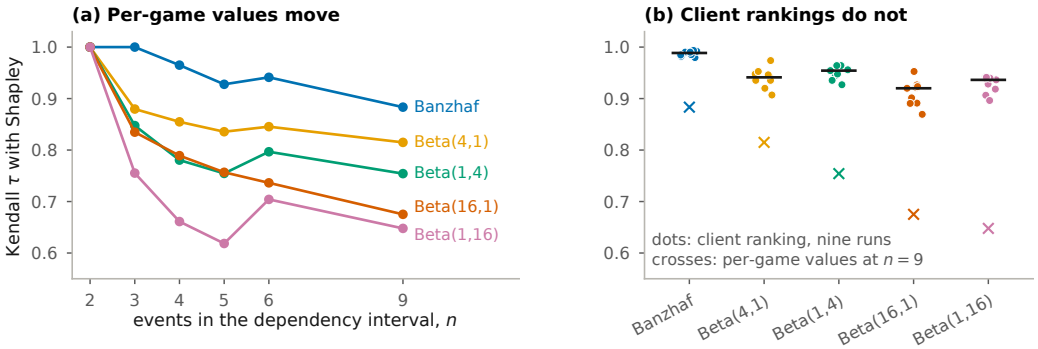

Figure 6: Sensitivity to the declared solution concept, recomputed exactly from complete coalition tables. (a) Mean Kendall correlation between each semivalue and the Shapley value within a game, by the number of events in the dependency interval. (b) Kendall correlation between the client ranking under each semivalue and under the Shapley value, one dot per run with a bar at the median. Crosses repeat the within-game value at nine events from (a). Values within a game move with the declared concept, while client rankings stay close to the Shapley ranking.

Table 9: Replacing the public set of the utility by a disjoint held-out set of 200 labelled samples, on 132 games with $n = 1$ to 12 from three CIFAR-10 development runs, twelve games per size. $D_{\pm}$ and ρ are averaged over sizes, and ρ omits $n = 1$. The first row measures how far the declared quantity moves with the data of its utility, and the second how far FedACE is from the quantity it declares.

Comparison	$D_{\pm} \downarrow$	$\rho \uparrow$	$t_l \uparrow$
Exact (public) vs exact (held-out)	.146	.890	.783
FedACE vs exact (public)	.036	.973	.932
FedACE vs exact (held-out)	.154	.879	.783

part of the change, yet the one-step gain still lowers the Kendall correlation to 0.918 on average and to 0.891 in the lowest run. Complete tables exist only up to nine events, so these figures leave out the long intervals where truncation removes the most. Table 11 holds every recorded update fixed and changes only the composition order. Reverse arrival order changes the event ranking in all 24 games and changes 84% to 91% of the composed coalition states.

Table 10: Truncating the dependency interval, computed exactly on the $20{,}488$ games with complete coalition tables. Keeping only the last k events and composing the older ones in factual order is, by Proposition 2, the game $S \mapsto \bar{v}_e(S \cup O)$ with O the $n-k$ oldest events. With $k=1$ the value reduces to the one-step gain $U_e(\theta_{e+1}) - U_e(\theta_e)$. Changed counts games with $n > k$, sign flips and $|\Delta\phi|$ are measured on those games, and client agreement compares rankings by $\bar{\Phi}_c$ with the declared ranking, as a mean over nine runs with the lowest run in parentheses.

		credited event		client ranking	
Events kept	Changed	sign flips	$\lvert\Delta\phi\rvert$	Kendall	Spearman ρ
full interval *(declared)*	0	—	—	1.000	1.000
last 1 event	14,779	12.1%	0.0469	0.918 (0.891)	0.985 (0.972)
last 2 events	7,919	9.6%	0.0349	0.949 (0.927)	0.993 (0.988)
last 3 events	3,626	7.2%	0.0256	0.963 (0.922)	0.994 (0.976)
last 4 events	1,819	4.5%	0.0136	0.978 (0.961)	0.998 (0.994)

Table 11: Changing only the composition order, with every recorded update held fixed, on the 24 event-level games of Figure 1b. They come from three CIFAR-10 runs with seeds 0 to 2, separate from the nine runs of Section 4 and trained under the same protocol. ρ is the Spearman correlation between event values under the alternative order and under factual arrival order, signed L_1 is the gap between the two value vectors, and states differing is the share of coalitions whose composed state changes. The last column counts games whose event ranking changes. Arrival order is the order the server applied, and Proposition 3 shows it is the only order consistent with the recorded prefixes under an explicit non-degeneracy condition.

n	Composition order	Games	ρ	signed L_1	states differing	ranking changed
5	reverse arrival	12	0.642	0.1414	83.9%	12/12
5	score-blind random	12	0.842	0.1108	66.7%	10/12
6	reverse arrival	12	0.671	0.1089	90.5%	12/12
6	score-blind random	12	0.843	0.0686	71.6%	11/12

B.7 Published evaluators

Each published evaluator runs as a passive observer on the recorded trajectory. Its adapter adds the per-event frozen snapshots of base model, local model, staleness and sample count needed to follow the common arrival order, and changes neither the valuation formula nor the server aggregation. Figure 7 shows the nine runs behind Table 12.

Table 12: Where published evaluators sit relative to the declared estimand, on the same nine recorded trajectories. Each entry is the Spearman correlation between a method's own client score and the client ranking induced by the exact value of the declared game, averaged over the nine runs with the standard error of that mean. A client score is read twice, once as a per-arrival mean and once as a cumulative total, because the published methods do not agree on which of the two they report. The last column counts runs with a positive correlation. Each method defines its own game or proxy, so an entry locates what the method measures and does not grade it. The last row calibrates the scale. The exact reference exists wherever a complete coalition table exists, which includes every game FedACE answers by enumeration, so that row mainly reflects which events the reference covers. Restricted to those events the FedACE ranking agrees with the reference at 0.9997. All scores come from passive replay and not from end-to-end deployment, where scores could alter the aggregation.

Method	Venue	per-arrival mean	cumulative	runs > 0
Proposed for asynchronous or dynamic participation				
AFedSV	Comput. Commun. 2026	$+0.263 \pm 0.103$	$+0.269 \pm 0.103$	$6/9$
FedDSV	IEEE TMC 2025	$+0.168 \pm 0.143$	$+0.045 \pm 0.125$	$6/9$
K-FL-BSV	IEEE BigData 2023	-0.310 ± 0.104	-0.310 ± 0.091	$2/9$
Proposed for synchronous rounds, adapted to the arrival window				
CGSV	NeurIPS 2021	$+0.411 \pm 0.052$	$+0.442 \pm 0.059$	$9/9$
ShapFed	IJCAI 2024	$+0.190 \pm 0.134$	-0.187 ± 0.049	$6/9$
SPACE	NeurIPS 2023	$+0.113 \pm 0.091$	-0.118 ± 0.109	$5/9$
TMC-Shapley	ICML 2019	$+0.005 \pm 0.139$	-0.089 ± 0.120	$3/9$
FedMS	TMLR 2025	-0.057 ± 0.096	-0.160 ± 0.092	$3/9$
Same-estimand solver, for calibration				
FedACE		$+0.969 \pm 0.007$	$+0.953 \pm 0.007$	$9/9$

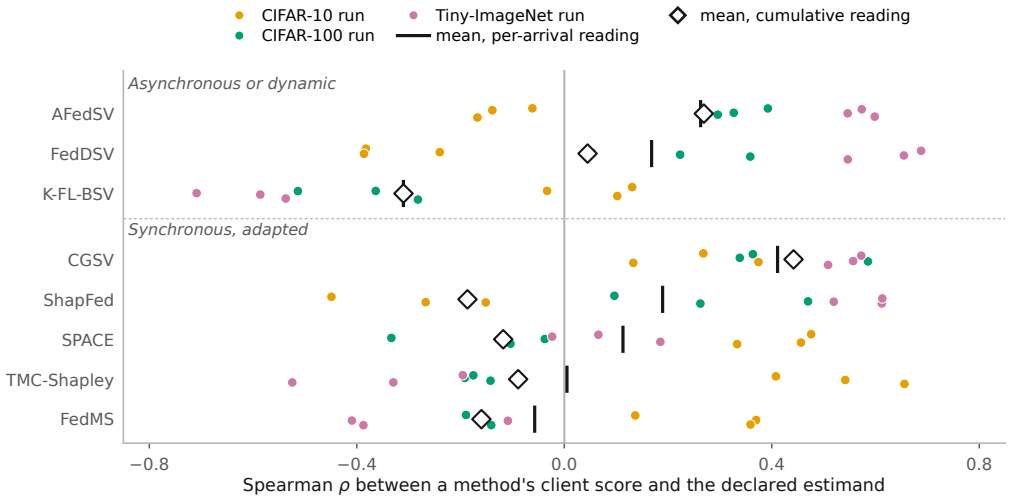

Figure 7: Agreement between each published evaluator and the declared estimand, one dot per run coloured by dataset. Bars mark the mean under the per-arrival reading and diamonds the mean under the cumulative reading. Only CGSV keeps one sign on all three datasets.

Table 13: Area under the ROC curve for separating the five corrupted clients from the 45 clean ones by client score, on CIFAR-10 with two training seeds. Values below 0.5 rank corrupted clients above clean ones. Fresh and stale rows restrict FedACE to arrivals with $\tau = 0$ and $\tau > 0$. Only six of the ten corrupted clients have a fresh arrival, so the fresh row rests on few clients.

	label noise		mixed attack	
Score	seed 0	seed 1	seed 0	seed 1
FedACE, all arrivals	.653	.556	.391	.382
FedACE, fresh arrivals only	.740	.667	.385	.500
FedACE, stale arrivals only	.658	.542	.387	.369
CGSV	.413	.598	.331	.476
TMC-Shapley	.413	.236	.342	.227
staleness only	.511	.544	.511	.544