Privacy-preserving Chunk Scheduling in a BitTorrent Implementation of Federated Learning

***Abstract*—Traditional federated learning (FL) relies on a central aggregator server, which can create performance bottlenecks and privacy risks. Decentralized *mix-and-forward* designs remove the server, but repeated local mixing can attenuate global information under heterogeneity and exposes peer-to-peer neighborhoods as a privacy attack surface. To preserve FedAvg-style aggregation semantics (over updates reconstructable by the round deadline) while scaling dissemination, we present *FLTorrent*, a BitTorrent-based dissemination layer for serverless FL with a short warm-up. Warm-up hardens *within-round source unlinkability*—a dissemination-layer goal orthogonal to content protections (e.g., DP or secure aggregation)—via (i) pre-round obfuscation, (ii) randomized lags, and (iii) coordination-only non-owner-first scheduling (tracker off the data path), before switching to vanilla BitTorrent swarming. We upper-bound the per-transfer attribution posterior by the fraction of owner chunks in a sender's eligible cover set, and derive a tighter high-probability bound that improves with early non-owner mass. A simple heuristic, GREEDYFASTESTFIRST, attains $\approx 92\%$ of a bandwidth-optimal max-flow upper bound, while warm-up remains a stable $\approx 12\%$ share of a round across 100–500 peers. Under an observation-only local adversary, FLTorrent drives attribution success close to neighborhood-level random guessing for typical nodes, improves with network size, and remains robust under collusion. In LLM-scale stress tests (Gemma-7B, DeepSeek-R1-14B, Qwen2.5-32B, and Llama-3.3-70B) over 7–$10\,\mathrm{Gbps}$ access links, FLTorrent adds only ~ 6–10% end-to-end overhead relative to BitTorrent-only. Overall, FLTorrent shows that within-round unlinkability and BitTorrent-level efficiency can co-exist with predictable, low overheads at scale.**

***Index Terms*—Federated learning, Peer-to-peer systems, BitTorrent swarming, Privacy, Unlinkability, Chunk Scheduling.**

I. INTRODUCTION

Federated Learning (FL) trains shared models without centralizing training data by exchanging model updates [1]. In the canonical server-based workflow, a central aggregator collects client updates, computes a weighted average (FedAvg), and redistributes the new model [1]. While widely adopted for privacy and compliance (e.g., GDPR), this architecture has its downsides: the server can become a communication bottleneck and requires full trust. These issues have motivated substantial distributed-systems work on communication-efficient aggregation, client selection, personalization, and robustness [2]–[5].

Despite keeping data local, FL is not immune to privacy inference. Prior work shows that adversaries can extract sensitive information from *update contents* via membership inference and gradient inversion/reconstruction [6]–[9]. Accordingly, many deployments adopt content-level defenses such as secure aggregation (and often DP), which limit visibility of individual updates to the aggregator [10]–[12]. However, such defenses do not address *metadata leakage* in the communication substrate—e.g., which participant likely produced which update and when—which becomes salient in peer-to-peer dissemination. This motivates reducing reliance on a single trusted aggregator *and* hardening the dissemination layer against within-round attribution.

A prominent direction is *decentralized learning* and decentralized FL (DL/DFL), where clients exchange information over an overlay graph and update their models via local mixing (e.g., push-sum/gossip, mixing matrices, or "mix-and-forward" propagation) [13]–[15]. This improves fault tolerance and reduces reliance on one aggregation point, but it complicates the "what is being computed" question and introduces proximity-driven privacy risks. First, local mixing is inherently a multi-hop approximation: within a finite time budget, each node only sees *partially mixed* information rather than the full per-round set of updates. Under heterogeneity (e.g., non-IID data, bandwidth skew, and churn), such repeated mixing can distort the effective aggregation and degrade convergence/robustness, which we refer to as *attenuation*. Second, peer-to-peer (P2P) proximity changes the privacy attack surface: neighbors can observe fine-grained protocol signals (e.g., timing and early-transfer patterns) and exploit proximity-driven privacy risks in decentralized ML settings [16]. Thus, existing decentralized designs often trade off among (i) FedAvg-style aggregation semantics, (ii) scalable dissemination, and (iii) robustness to proximity-driven privacy threats.

Main idea. To avoid semantic attenuation while scaling dissemination, we build the communication substrate on BitTorrent [17]. Rather than repeatedly mixing models over hops, we treat each client's update as a file to be disseminated: the update is split into fixed-size chunks, and peers exchange chunks in parallel within a swarm. Swarming amortizes upload cost (others replicate what you seed), leverages heterogeneous bandwidth, and sustains high utilization via tit-for-tat incentives [17], [18]. Most importantly, *if each round's updates are fully disseminated (i.e., reconstructable) by the round deadline*, then every client locally computes the *same* FedAvg aggregate as in server-based FL, without a central aggregator [1]. When some updates are not reconstructable by the deadline (e.g., due to partial participation or timeouts), FLTorrent computes FedAvg over the reconstructable active set, matching the standard FL treatment of missing updates. In other words, BitTorrent aligns dissemination with aggregation semantics: once peers obtain the same set of per-round updates by the deadline, they naturally agree on the same aggregate.

However, vanilla BitTorrent optimizes throughput and avail-

ability, not privacy. In P2P FL, this mismatch creates a protocol-specific privacy objective: *within-round source unlinkability* (attribution resistance) between a client and its update at the P2P layer. Concretely, at round start, each client initially holds only its own chunks; consequently, early transmissions are biased toward "owner" chunks, and a neighbor can attribute an update by correlating which chunks it first observes from which sender. Such attribution is not merely cosmetic: once an update can be linked to a specific client or silo (e.g., a hospital), the consequences become materially worse and can be actionable (e.g., revealing spatio-temporal operational signals). For example, *source inference attacks* aim to identify *which client* contributed a particular training record; Hu *et al.* [19] show that identifying the *source hospital* can already be actionable (e.g., revealing whether a hospital is in a high-risk region), and Chen *et al.* [20] discuss that spatio-temporal source inference can reveal sensitive dynamics (e.g., during a rare disease outbreak). Even when secure aggregation hides individual updates from the server, it does not preclude inference about participant-level attributes (e.g., client quality) from aggregated outputs [21]. In this paper, we focus on local, observation-only adversaries within the swarm and on permissioned settings where admission control mitigates Sybil attacks; global traffic analysis is out of scope.

Contributions. We introduce **FLTorrent**, a BitTorrent-based dissemination layer with a short privacy-preserving warm-up phase that hardens the protocol against *within-round source attribution* while preserving FedAvg-style aggregation semantics over the reconstructable set by the round's deadline. FLTorrent combines a short privacy-preserving warm-up phase with standard BitTorrent swarming: during warm-up, it injects uncertainty about chunk origin via (i) *pre-round obfuscation* (injecting limited non-owner chunk mass before time 0), (ii) *time obfuscation* (random lags that blur temporal correlation), and (iii) *non-owner-first scheduling* coordinated by a tracker (coordination-only and not on the data path) to avoid early, owner-revealing transfers. Warm-up ends once each peer has accumulated at least k chunks in its cover set; at that point, per-transfer attribution confidence is provably capped by a term that decreases on the order of $1/k$ (details in our analysis), after which FLTorrent switches to vanilla BitTorrent for efficient bulk dissemination. We further (i) formalize within-round attribution in the P2P setting and derive unlinkability bounds, (ii) characterize a max-flow-style upper bound for bandwidth-optimal warm-up scheduling and develop practical heuristics, and (iii) release FLTorrent as an open-source framework to facilitate reproducibility and follow-on research.[1]

Results. Our greedy warm-up scheduler, GREEDYFASTEST-FIRST, attains $\approx 92\%$ of the max-flow-style upper bound. Across 100–500 peers, warm-up remains a stable $\approx 11.5\%$–12.4% fraction of a round with overall utilization 75%–80%. With all defenses enabled, the maximum attribution success rate approaches random guessing (approximately $1/m$ under default connectivity). In LLM-scale stress tests (Gemma-7B, DeepSeek-R1-14B, Qwen2.5-32B, Llama-3.3-70B) over 7–10 Gbps links, FLTorrent adds only $\sim 6\%$–10% round-time overhead relative to BitTorrent-only.

[1]Code: https://anonymous.4open.science/r/FLTorrent-anonymous-D8EF

II. PROBLEM FORMULATION

A. Background: BitTorrent primitives and FL rounds.

BitTorrent [17] disseminates an object by splitting it into fixed-size chunks (BitTorrent *pieces*) and exchanging them in parallel among peers. A *torrent descriptor* (metadata) enumerates chunk identifiers (piece indices) and cryptographic hashes, enabling receivers to verify chunk integrity and discard corrupted payloads [22]. Peers advertise availability via *bitfields* and use swarming incentives (e.g., tit-for-tat) to sustain high utilization [17]. In vanilla BitTorrent, the tracker mainly supports peer discovery and does not schedule chunks. In *FLTorrent*, the tracker additionally collects *per-peer chunk availability* (bitfields) during a short warm-up period and issues early scheduling directives; it is not on the data path and does not need access to chunk contents. A BitTorrent *peer* corresponds to an FL *client*; we use "client" throughout and use "peer" only when referring to BitTorrent mechanisms.

B. System Model (One Round)

We consider a BitTorrent-style tracker $\mathcal{T}$ and a set of clients $\mathcal{V}$ ($|\mathcal{V}| = n$). For a single FL round $r \in \mathbb{N}$, the tracker forms an overlay graph $G^r = (\mathcal{V}, \mathcal{E}^r)$ and coordinates warm-up scheduling using the bitfields it collects. For $v \in \mathcal{V}$, its neighborhood is $\mathcal{N}^r(v) = \{w : (v,w) \in \mathcal{E}^r\}$ and the average degree is $m = \frac{1}{n}\sum_{v\in\mathcal{V}} |\mathcal{N}^r(v)|$. Time is slotted with index $s \in \{0, 1, \dots\}$ and duration $\Delta = 1$. Each round has a deadline at slot $s_{\max}$ (used throughout for warm-up termination and aggregation). Each client v has uplink/downlink capacities U_v, D_v (bytes/slot). Under a dissemination/scheduling *policy* π, instantaneous rates $u_v[s;\pi]$ and $d_v[s;\pi]$ satisfy $0 \le u_v[s;\pi] \le U_v$, $0 \le d_v[s;\pi] \le D_v$, and flow conservation $\sum_{v\in\mathcal{V}} u_v[s;\pi] = \sum_{v\in\mathcal{V}} d_v[s;\pi]$ for all s. When convenient, we convert byte budgets into per-slot chunk budgets as $u_v = \lfloor U_v\Delta/C \rfloor$ and $d_v = \lfloor D_v\Delta/C \rfloor$.

Client v produces update g_v^r of size S_v^r bytes, sliced into $K_v^r = \lceil S_v^r/C \rceil$ chunks of C bytes each. For exposition, we label these chunks as $\mathcal{C}_v^r = \{(v,r,i) : i = 1, \dots, K_v^r\}$ and $\mathcal{C}^r = \bigcup_{v\in\mathcal{V}} \mathcal{C}_v^r$. (These tuples are *analysis labels*; in the actual protocol, peers exchange a torrent identifier and a piece index, which do not explicitly encode v.) For any client $v \in \mathcal{V}$, let $\mathcal{C}_v^r[s] \subseteq \mathcal{C}^r$ denote the chunks held by v at slot s. The (full) completion time under policy π (assuming all clients remain active) is $T_{\text{full}}(\pi) = \min\{s : \forall v \in \mathcal{V}, \mathcal{C}_v^r[s] = \mathcal{C}^r\}$.

Aggregation semantics. At round start, each update is published with a torrent descriptor and a scalar weight (e.g., local sample count). In our analysis, we assume homogeneous update sizes across clients (i.e., $S_v^r = S$ for all v and thus $K_v^r = K$); consequently, torrent descriptors—which contain only chunk hashes and piece counts—do not reveal owner identity.[2] Peers learn only which chunks form a given update

[2]When update sizes vary significantly, an attacker could narrow down candidate owners by matching observed chunk counts to known update sizes. Mitigations include padding updates to a common size or grouping clients with similar update sizes; we leave such extensions to future work.

and its associated weight, not which client produced it. By the round deadline $s_{\max}$, each client v reconstructs the subset of updates whose chunks it has obtained and locally computes a FedAvg aggregate over its *reconstructable set*

$$g_v^{\mathrm{agg},r} = \sum_{u \in \mathcal{A}_v^r} \frac{w_u}{\sum_{j \in \mathcal{A}_v^r} w_j} g_u^r, \mathcal{A}_v^r := \{u \in \mathcal{V} : \mathcal{C}_u^r \subseteq \mathcal{C}_v^r[s_{\max}]\}$$

with $|\mathcal{A}_v^r| \geq 1$ required for aggregation. (Indexing by client identity u is for analysis; in the protocol, peers identify updates by their published descriptors/pseudonyms, not by real identities.) Under within-round dropouts, $\mathcal{A}_v^r$ excludes updates that v cannot reconstruct by $s_{\max}$; a client may disconnect after seeding chunks, yet its update can still belong to $\mathcal{A}_v^r$ if chunks have been sufficiently replicated.

Observations. Let $\mathcal{O}_v[s]$ denote client v's observations up to slot s: received piece indices (chunk identifiers), sender round-pseudonyms, timestamps/order, and (during warm-up) tracker coordination messages it receives. Round pseudonyms are *stable within a round* (accounting/tit-for-tat) and *rotate across rounds* to mitigate cross-round linkability.

C. Threat Model

Goal. We target *peer-side within-round source unlinkability* (§II-D) against inference from proximity-visible metadata. Because clients publish torrent descriptors, the tracker inherently learns the chunk-to-owner mapping; hiding it from the tracker is *not* our objective.

Channels and scope. We assume secure and authenticated channels (e.g., TLS), which protect contents but not traffic metadata (endpoints/timing/volume); thus Internet-scale traffic analysis remains out of scope. We assume round pseudonyms are tracker-issued and authenticated over these channels; spoofing another client's pseudonym reduces to credential compromise or Sybil admission and is out of scope.

Adversary A (HBC peers). A static attacker corrupts a subset of clients before a round begins. Corrupted clients follow the protocol but infer attribution from (i) chunk indices received from each neighbor (identified by sender pseudonym), (ii) transfer timing/ordering, and (iii) warm-up directives received from the tracker. They may collude by pooling observations. Our attacks in §IV-C instantiate this adversary.

Adversary B (Byzantine peers). We additionally consider Byzantine clients that may deviate arbitrarily (e.g., lie in bitfields, selectively forward/withhold chunks, or delay transmissions). Payload tampering is detectable via chunk-hash verification from the torrent descriptor [22]; thus undetectable content manipulation is out of scope. Our unlinkability bounds (§IV-A) apply to transfers *sent by honest peers*; liveness under deviations is handled operationally (§III-E).

Tracker integrity assumption. The tracker coordinates overlay formation and warm-up directives but is not on the data path. We assume a trusted tracker by default; an optional auditable (accountability) variant is described in §III-D. Tracker–client collusion is excluded since the tracker knows the chunk-to-owner mapping from descriptors and collusion would trivially reveal attribution.

Out of scope. (i) Global network adversaries (traffic-pattern analysis; defense requires anonymous routing, orthogonal to our focus). (ii) Sybil attacks (we assume permissioned membership). (iii) Denial-of-service (e.g., forcing timeouts/fail-open) beyond our operational handling. (iv) Update-content attacks (complementary to secure aggregation/DP).

D. Privacy Objective and Metrics

Within-round source unlinkability. Under the above adversaries, BitTorrent-style dissemination introduces an attribution channel: early in a round, a sender is more likely to transmit its *own* chunks, enabling neighbors to infer sender–source associations. For any observed transfer of chunk c, let $\mathsf{snd}(c)$ be the sender (round-pseudonym) and $\mathsf{src}(c)$ the true source client of c. We say the system attains **unlinkability level** P **at slot** s if for all observers v and observed transfers, $\Pr[\mathsf{src}(c) = \mathsf{snd}(c) | \mathcal{O}_v[s]] \leq 1/P$. Let $T_P(\pi) = \min\{s :$ unlinkability level P holds$\}$ be the time to reach level P.

Warm-up knob k (cover-set threshold). FLTorrent uses a short warm-up phase and switches to vanilla BitTorrent once each peer has accumulated at least k chunks. Here, k is a *mechanism parameter* controlling the sender's minimum *cover set* (inventory) size before high-rate swarming begins. In our experiments, we report the threshold as a percentage $\alpha = k/K$ of a *single update*'s chunk count (e.g., $\alpha = 10\%$); under homogeneous update sizes, $K_v^r = K$ for all v and $k = \lceil \alpha K \rceil$ (e.g., $K \approx 206$ for a 51 MB update with 256 KB chunks, so $k \approx 21$ at $\alpha = 10\%$). Our analysis (Section IV-A) relates this cover-set threshold to an attribution-posterior cap: for any honest sender, the per-transfer posterior is bounded by κ_u/k (and tightens as early non-owner mass increases).

Efficiency metrics. Over H slots, aggregate throughput and normalized utilization are

$$\bar{U}(\pi; H) = \tfrac{1}{H} \textstyle\sum_{s,v} u_v[s; \pi],$$
$$\mathrm{Util}(\pi; H) = \bar{U}(\pi; H) \big/ \textstyle\sum_v U_v \in [0, 1].$$

Goal. We seek policies π that (i) reach a target unlinkability level quickly (small T_P), (ii) maintain high utilization, and (iii) minimize completion time T_{full}.

III. *FLTorrent* Design

We present *FLTorrent* as a BitTorrent-based dissemination framework for serverless aggregation, with a warm-up phase that hardens the dissemination layer against within-round source attribution. Section II defines the system and threat model , and our unlinkability objective; this section focuses on how it works.

A. Overview and Round Workflow

FLTorrent runs in synchronous FL rounds. In each round r, clients disseminate their local updates via a BitTorrent-style swarm, with an explicit warm-up that hardens the dissemination layer against *within-round source attribution*. Each round consists of:

(1) **Local training**—each client v computes a local update g_v^r from its private data;

(2) **Chunking & metadata publication**—v splits g_v^r into fixed-size chunks and publishes a torrent descriptor desc_v^r

TABLE I: Notation summary.

Sym.	Meaning	Sym.	Meaning
System & Network			
$\mathcal{T}$	Tracker	$\mathcal{V}$	n clients
G^r	Overlay $(\mathcal{V}, \mathcal{E}^r)$	$\mathcal{N}^r(v)$	Neighbors of v
m	Avg. degree	r	FL round index
s, Δ	Slot index, duration	$s_{\max}$	Round deadline
s_{BT}	BT-phase start	U_v, D_v	Up/down capacity
u_v, d_v	Per-slot chunk budget	C	Chunk size
Updates & Chunks			
g_v^r, S_v^r	Update, size	K_v^r	#chunks of g_v^r
$\mathcal{C}_v^r$	Chunk IDs of g_v^r	$\mathcal{C}^r$	$\bigcup_v \mathcal{C}_v^r$
$\mathcal{C}_v^r[s]$	Chunks held at slot s		
Warm-up & Scheduling			
π	Dissem. policy	R, σ	Spray ratio, copies
k, α	Cover-set threshold, % of K	$T_{\mathrm{lag}}, \ell_v^r$	Lag window, start lag
Analysis & Privacy			
$\mathcal{B}_u^r[s], B_u$	Eligible buffer, size	κ_u, O_u	Owner throttle, count
X_u	Non-owner count	Z_R, Z_T	Non-owner terms
$\mathcal{O}_v[s]$	Observations	$\mathsf{snd}, \mathsf{src}$	Sender, source
P, T_P	Unlinkability level, time	ρ_u, ϕ	Coalition frac., strength
Aggregation			
$\mathcal{A}_v^r$	Reconstructable set		

(with per-chunk hashes [22]) and a round pseudonym pid_v^r (stable within-round, rotated across rounds);
(3) **Warm-up**—the tracker coordinates early transfers so each active client accumulates at least k chunks before normal BitTorrent announcements dominate;
(4) **BitTorrent swarming**—clients switch to vanilla BitTorrent to fetch remaining chunks;
(5) **Aggregation**—by the deadline, each client performs FedAvg over its reconstructable set $\mathcal{A}_v^r$ (Section II-B);
(6) **Audit (optional)**—under the auditable tracker model, the tracker reveals per-round randomness and logs for post-hoc verification (Section III-D).

We parameterize k relative to a single update's chunk count (e.g., $k = \lceil \alpha K \rceil$ with $\alpha = 10\%$ under homogeneous update sizes), consistent with Section II-D.

Protocol skeleton. At round r, each client $v \in \mathcal{V}$ publishes $(\mathsf{desc}_v^r, \mathsf{pid}_v^r)$, where desc_v^r defines its chunk set $\mathcal{C}_v^r$ and per-chunk hashes. The tracker distributes all published descriptors, samples an overlay G^r, and (optionally) executes a one-shot pre-round spray that seeds a small fraction of chunks to random non-neighbors via ephemeral tunnels. During warm-up ($s < s_{\mathrm{BT}}$), the tracker uses collected bitfields to issue stage schedules $\mathsf{Sched}^r[s]$ satisfying feasibility constraints (adjacency, availability, per-stage capacity caps), preferring non-owner senders when available. Warm-up ends at $s_{\mathrm{BT}} = \min\{s : \forall v \in \mathcal{V}_{\mathrm{active}}^r[s], |\mathcal{C}_v^r[s]| \geq k\} \wedge s_{\max}$, after which clients run vanilla BitTorrent until $s_{\max}$. A peer attacker at client w observes sender pseudonyms, chunk indices, and transfer timing throughout the round. At the deadline, each client aggregates over its reconstructable set $\mathcal{A}_v^r$ (Section II-B).

B. Warm-Up Phase for Source Unlinkability

Why warm-up is necessary. At round start, each client holds only its own local chunks. If clients immediately request a vanilla BitTorrent announcement, early transfers are biased toward "owner to neighbors," making attribution easy. Warm-up mitigates this by ensuring each sender has a sufficiently large *cover set* before normal BitTorrent dynamics dominate.

Warm-up knob k and unlinkability level. We use k as a *mechanism parameter*: warm-up ends once each active client has accumulated at least k chunks (we set $k = \lceil \alpha K \rceil$ as a fraction α of a single update's chunk count K under homogeneous update sizes). Intuitively, when a sender chooses which chunk to transmit from a larger inventory, the probability that an observed early transfer reveals the sender as the source decreases. Section IV-A formalizes this as a $1/P$-type posterior bound, where the effective unlinkability level P increases with k (and with the strength of obfuscation).

1) Pre-round Obfuscation Phase Pre-round obfuscation occurs before the round begins and injects a small amount of non-owner chunk mass into the system. We parameterize its strength by a *ratio* $R \in (0, 1)$ (or R in %): for each source client v with K_v^r chunks, the tracker randomly selects $\lfloor R \cdot K_v^r \rfloor$ *chunk identifiers* from $\mathcal{C}_v^r$ and assigns each selected identifier to a randomly chosen *non-neighbor* recipient. To execute a spray transfer, the tracker sends control instructions (containing the chunk identifier) to both the owner v and the designated recipient, directing them to establish an ephemeral direct tunnel, perform exactly one chunk transmission, and then tear the tunnel down. Crucially, the tracker operates only on chunk identifiers and client identifiers—it never receives or forwards chunk payloads, and observes completion acknowledgments. We choose a small R (default $R = 0.2$) to balance the injected non-owner mass against control-plane coordination overhead.

2) Time Obfuscation Pre-round obfuscation alone is insufficient if all clients start transmitting at the same time: receivers can still correlate first-arrival times with sender ownership, yielding strong within-round attribution signals. We therefore add randomized start-time lags. At the beginning of each round, each client v samples a lag $\ell_v \sim \mathrm{Unif}\{0, 1, \ldots, T_{\mathrm{lag}} - 1\}$ and delays initiating transmissions by ℓ_v slots. This blurs temporal correlation and increases the likelihood that early transfers are relayed (non-owner) rather than owner chunks. However, time obfuscation can waste uplink capacity during idle waits, motivating tracker-assisted scheduling below. In our default configuration, $\Delta = 1\,\mathrm{s}$ and $T_{\mathrm{lag}} = 3$, i.e., $\ell_v \sim \mathrm{Unif}\{0, 1, 2\}$ seconds.

3) Tracker-assisted Non-owner-first Scheduling Pre-round and time obfuscation improve unlinkability but can add coordination overhead (spray) and/or transient under-utilization (lags). To reach the target unlinkability level quickly *without* sacrificing throughput, FLTorrent lets the tracker coordinate warm-up transfers. During warm-up, the tracker collects per-client bitfields and assigns transmissions so that, whenever possible, a requested chunk is served by a *non-owner* holder (a client that previously obtained the chunk), rather than by its original source. This *non-owner-first* preference reduces early owner bias: once a sender has accumulated sufficient non-local chunks, an observer cannot treat "sender" as a reliable proxy for "source." Formally, in a warm-up stage starting at slot $s < s_{\mathrm{BT}}$, the tracker knows each client's inventory $\mathcal{C}_v^r[s]$ and its missing-chunk set $\mathcal{M}_v^r[s] = \left(\bigcup_{u \in \mathcal{N}^r(v)} \mathcal{C}_u^r[s]\right) \setminus \mathcal{C}_v^r[s]$. It schedules feasible

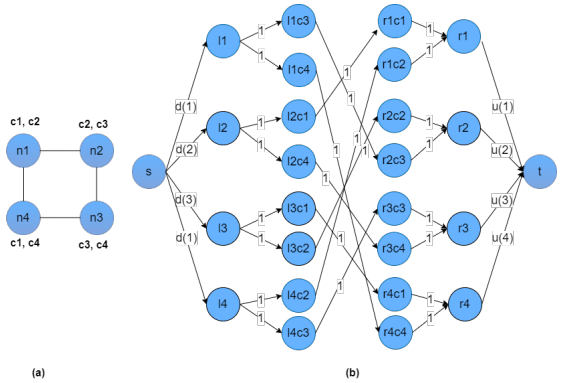

Fig. 1: a) Example overlay for a group with 4 nodes and 4 chunks. Each node is depicted with the chunks it holds. b) The resulting max-flow network.

missing-chunk transmissions (respecting per-stage capacity and avoiding duplicates) until all *active* clients reach the k-chunk cover-set threshold (or until $s_{\max}$); at s_{BT} FLTorrent switches to vanilla BitTorrent.

Lemma 1 (Hardness of optimal warm-up scheduling). Minimizing the warm-up makespan (time until all clients have at least k chunks) is NP-complete, by a standard reduction from precedence-constrained makespan scheduling (e.g., $P \mid \mathrm{prec} \mid C_{\max}$) [23], [24].

Accordingly, we use tractable heuristics that aim to maximize warm-up bandwidth utilization, which empirically yields low makespan in our settings.

C. Warm-up Scheduling

1) Bandwidth-optimal Scheduling as a Max-flow Upper Bound: A bandwidth-optimal *stage-wise* warm-up schedule maximizes the group throughput within a stage (sum of used uplink, equivalently downlink), given current inventories and per-stage chunk budgets. We schedule in discrete stages of duration Δ. Each client v can upload at most $u_v = \lfloor U_v \Delta / C \rfloor$ chunks and download at most $d_v = \lfloor D_v \Delta / C \rfloor$ chunks per stage. Stage-wise throughput maximization reduces to a max-flow over a bipartite construction with supply u_v and demand d_v, plus availability edges ensuring a chunk is only sent by a neighbor that already possesses it. Figure 1 illustrates an example topology and its corresponding max-flow network. We do not run max-flow online; we use it only as an (offline) upper bound computed with full knowledge of the stage state.

2) Practical Schedulers: Since max-flow is impractical at scale, we implement centralized and distributed heuristics, as well as flooding-style dissemination.

3) Centralized RandomFIFO: The tracker assigns each missing-chunk request to a random client that holds the chunk. Clients serve assigned chunks FIFO. A client can send up to τ chunks simultaneously (BitTorrent commonly uses $\tau = 4$).

4) Centralized RandomFastestFirst: Like RandomFIFO, but a sender prioritizes the τ fastest requesters based on available sender uplink and available downlink at the requester.

5) Centralized GreedyFastestFirst: Like RandomFastestFirst, but the tracker assigns each missing-chunk request to the fastest feasible sender.

6) Distributed Scheduling (Neighborhood-level announcements): In distributed scheduling, clients decide which requests to issue and who to serve. To avoid revealing which specific neighbor holds which chunk during warm-up, the tracker can provide neighborhood-level availability: for each client v, it announces $\mathcal{C}^{\mathrm{TA}}(v,s) = \bigcup_{u \in \mathcal{N}^r(v)} \mathcal{C}_u^r[s]$, without disclosing the exact mapping chunk $\rightarrow$ neighbor.

7) Flooding: In flooding, each client sends chunks to neighbors randomly without repetition. Combined with randomized lags, flooding increases relaying likelihood but wastes bandwidth and performs worse than coordinated warm-up heuristics.

Non-owner-first refinement. When multiple feasible senders can serve a missing chunk, we apply a non-owner-first tie-breaking preference to the centralized schedulers: the tracker prioritizes a sender that is not the original source of that chunk; otherwise it falls back to the source.

D. Auditability Under a Deviating Tracker

Motivation. Although the tracker is not a privacy target (torrent descriptors reveal the chunk-to-owner mapping), a deviating tracker can still *amplify peer-side attribution* by shaping the environment—e.g., generating sparse/imbalanced overlays or issuing warm-up directives that slow early mixing. This matters in permissioned cross-silo consortia where a biased or faulty operator may *selectively* weaken some participants' unlinkability, increasing the success probability of *any* observation-only peer attacker (honest-but-curious neighbors), even without tracker–client collusion. In permissioned consortia, this mechanism supports external governance; engineering extensions such as replicated trackers or threshold-controlled coordination are complementary and left to future work.

Commit-then-reveal accountability. Before seeing per-round inputs, the tracker commits to a public seed $h^r = H(\mathsf{seed}^r)$; after the round it reveals seed^r and a log Log^r (overlay and warm-up directives). Clients can recompute the overlay and verify hard constraints (adjacency, configured per-stage caps, and no redundant deliveries except logged retries); otherwise they fail open to vanilla BitTorrent and treat unlinkability guarantees for that round as void.

E. Fault Tolerance in FLTorrent

BitTorrent swarming is naturally robust to intermittent connectivity and heterogeneous bandwidth once multiple replicas exist. Thus, FLTorrent's key reliability objective is to safely transition through warm-up; after that, BitTorrent provides standard resilience mechanisms (retries, resumption, multi-source downloads, tit-for-tat, dynamic choking).

Within-round dropouts. If a client disconnects during a round, the tracker excludes it from *further* scheduling decisions (it no longer issues or serves warm-up assignments). If the client had already uploaded chunks that have been replicated, dissemination can still complete for remaining participants, and the disconnected client's update may remain in the active set if it is reconstructable by the deadline. If the disconnected client was the sole holder of some chunks (disconnects before replication), those chunks become unavail-

able; the round completes over the remaining active set, as in partial participation in practical FL deployments. Disconnected clients can rejoin in the next round with fresh pseudonyms.

Cross-round churn (joins/leaves). Joiners are incorporated at the next round boundary. Persistent leavers are removed from the next round's overlay generation. The tracker regenerates a fresh overlay each round, preventing long-lived neighbor relationships that could amplify cross-round linkage.

Byzantine behavior handling. We distinguish (a) *integrity-violating* deviations and (b) *liveness-degrading* deviations. For (a), invalid-chunk injection or payload tampering is detectable via descriptor hash checks [22] and such content is discarded; under authenticated channels, spoofing another client's round pseudonym is excluded (otherwise it reduces to Sybil/credential compromise as in §II-C). For (b), Byzantine peers may lie in bitfields, selectively forward/withhold chunks, or delay transmissions to slow warm-up. These behaviors primarily impact efficiency/liveness rather than enabling stealthy attribution increases for transfers *sent by honest peers* that enforce eligibility rules; accordingly, our unlinkability bounds apply to honest senders. Operationally, we mitigate slowdown by (i) per-peer progress timeouts (clients that fail to make progress are marked inactive), and (ii) evaluating warm-up completion over the remaining active set. If warm-up cannot complete within $s_{\max}$, the system fails open to vanilla BitTorrent, preserving liveness but voiding unlinkability guarantees for that round (a DoS objective beyond our formal scope).

Privacy implications. Because unlinkability guarantees rely on cover-set size and non-owner mass, we evaluate the warm-up threshold over the currently active participants. A dropout reduces available capacity but does not invalidate unlinkability for other clients, as long as the warm-up threshold is satisfied for the remaining active set.

IV. Unlinkability Analysis and Attacks

A. Adversary Success Bounds

Setting and objective. We analyze *within-round source unlinkability* defined in Section II-D under the threat model in Section II-C. Concretely, an adversary (possibly a coalition of clients) observes protocol-level signals $\mathcal{O}_v[s]$ (sender round-pseudonyms, piece indices (chunk identifiers), timestamps/order, and warm-up scheduling directives received by v) and attempts to attribute an observed transfer to its true source. We focus on the within-round attribution event $\mathsf{src}(c) = \mathsf{snd}(c)$ for an observed chunk transfer of c. Unless stated otherwise, the bounds below apply to transfers sent by *honest senders* under origin-oblivious chunk selection (e.g., uniform or rarest-first independent of origin), consistent with Adversary A (and the honest-sender portion of Adversary B).

Eligible serving buffer and knobs. A client always *possesses* its own update chunks locally, but FLTorrent controls which chunks are *eligible for upload* during warm-up. For a sender u at a serving instant (round r, slot s), let $\mathcal{B}_u^r[s]$ be the set of currently eligible chunks for upload, and let $B_u := |\mathcal{B}_u^r[s]|$. The buffer contains (i) *non-owner* chunks received from other clients and (ii) a limited number of *owner* chunks from u's update. Let O_u be the number of eligible owner chunks and $X_u := B_u - O_u$ the eligible non-owner mass at that instant.

Notation clarification. Throughout this section, u denotes the *sender* (the client transmitting a chunk) and v denotes the *receiver/observer* (the client receiving the chunk and potentially acting as an attacker), matching $\mathcal{O}_v[s]$.

FLTorrent uses two enforcement knobs during warm-up: (1) **cover-set gating:** an honest sender enables any owner chunk only after $B_u \geq k$, and (2) **owner throttling:** at any time $O_u \leq \kappa_u$ (default $\kappa_u = 1$). When serving, honest senders apply an *origin-oblivious* chunk-selection rule: conditioned on $\mathcal{B}_u^r[s]$, the rule does not explicitly privilege owner chunks beyond what is implied by availability/state (e.g., uniform-at-random over $\mathcal{B}_u^r[s]$, or rarest-first restricted to $\mathcal{B}_u^r[s]$).

Per-transfer posterior and the $1/P$-type bound. Consider a transfer $u \to v$ of chunk c observed by v at some warm-up time. Under origin-oblivious selection, the (per-transfer) attribution posterior equals the eligible owner fraction: $\Pr[\mathsf{src}(c){=}u \mid \mathcal{O}_v] = O_u/B_u$. By cover-set gating ($B_u \geq k$) and throttling ($O_u \leq \kappa_u$), we obtain:

$$\Pr\big[\mathsf{src}(c){=}u \mid \mathcal{O}_v\big] = \tfrac{O_u}{B_u} \leq \tfrac{\kappa_u}{k}. \tag{1}$$

With $\kappa_u{=}1$, each warm-up transfer from an honest sender satisfies $\Pr(\text{correct attribution}) \leq 1/k$, matching the $1/P$-style objective in §II-D with $P \geq k/\kappa_u$. For intuition, a neighborhood-level "random guessing" baseline corresponds to posteriors on the order of $1/|\mathcal{N}^r(v)|$ for an observer v; our goal is to drive per-transfer attribution toward this regime while keeping warm-up short.

Warm-up mixing further tightens the posterior (high probability). Warm-up increases the eligible non-owner mass X_u, making O_u/B_u typically smaller than the worst-case cap κ_u/k. Write $X_u = Z_R(u) + Z_T(u)$, where $Z_R(u)$ is injected by pre-round spray and $Z_T(u)$ is gained early via randomized lags and non-owner-first scheduling.

Pre-round obfuscation. Each sender sprays $\sigma = \lfloor R \cdot K_u^r \rfloor$ chunks to uniformly random non-neighbors. For a fixed u, let $\mu_u := \mathbb{E}[Z_R(u)]$; under random assignment,

$$\mu_u = \textstyle\sum_{v:\, u \notin \mathcal{N}^r(v) \cup \{v\}} \frac{\sigma}{n-1-|\mathcal{N}^r(v)|},$$

so $\mu_u = \sigma$ for an m-regular overlay and $\mu_u \approx \sigma$ for near-regular overlays. Since $Z_R(u)$ is a Poisson-binomial sum, a Chernoff-type bound yields $\Pr[Z_R(u) \leq (1{-}\epsilon)\mu_u] \leq \exp(-\epsilon^2\mu_u/2)$ for $\epsilon \in (0,1)$.

Time obfuscation and non-owner-first scheduling. If lags are i.i.d. uniform on $\{0,\ldots,T_{\text{lag}}{-}1\}$, then $p_{\text{lead}} = \Pr[\ell_v < \ell_u] = (T_{\text{lag}}{-}1)/(2T_{\text{lag}})$. With average degree m and availability factor $q \in (0,1]$, we have $\mathbb{E}[Z_T(u)] \geq m\, p_{\text{lead}}\, q$ and $\Pr[Z_T(u) \leq (1{-}\epsilon)\mathbb{E}[Z_T(u)]] \leq \exp(-\epsilon^2\mathbb{E}[Z_T(u)]/2)$.

By a union bound, with probability at least $1 - \eta$ where $\eta = \exp(-\epsilon^2\mu_u/2) + \exp(-\epsilon^2\mathbb{E}[Z_T(u)]/2)$, we have $X_u \geq (1{-}\epsilon)\big(\mu_u + m\frac{T_{\text{lag}}-1}{2T_{\text{lag}}}q\big)$, hence

$$\tfrac{O_u}{B_u} = \tfrac{O_u}{O_u+X_u} \leq \frac{\kappa_u}{\kappa_u + (1-\epsilon)\big(\mu_u + m\frac{T_{\text{lag}}-1}{2T_{\text{lag}}}q\big)}. \tag{2}$$

Thus, warm-up mechanisms never violate the cap κ_u/k in Eq. (1) and typically tighten it further.

Conditionality under an auditable tracker. Under the *auditable tracker* model in §II-C, the bounds above are conditional on the published round log passing verification of the verifiable constraints in §III-D. If violations are detected, clients fail open to vanilla BitTorrent and treat unlinkability guarantees for that round as void.

B. Collusion-aware Bounds (Alliance Filtering)

We incorporate a strong coalition that can recognize and discount some non-owner traffic originating from coalition members. Fix an honest sender u at a serving instant. Let $\rho_u \in [0,1]$ denote the fraction of X_u originating from the coalition, and let $\phi \in [0,1]$ be the coalition's *recognition strength* (ϕ=0: no filtering; ϕ=1: perfect erasure). We treat ϕ as an analysis parameter capturing coalition side information; in evaluation we report envelopes spanning $\phi \in \{0,1\}$ to bracket practical recognition strength. Alliance filtering reduces effective non-owner mass to $X_u^{\text{eff}} = (1-\phi\rho_u)X_u$, giving

$$\theta_u^{\text{AF}} = \frac{O_u}{O_u+X_u^{\text{eff}}} \le \min\Big\{\frac{\kappa_u}{k},\ \frac{\kappa_u}{\kappa_u+(1-\phi\rho_u)X_u}\Big\}. \tag{3}$$

Coalition filtering loosens mixing by $(1-\phi\rho_u)$ but cannot beat the gating cap κ_u/k for honest senders.

High-probability mixing under collusion. Let $X_u = Z_R(u) + Z_T(u)$ as before. With probability at least $1-\eta$,

$$\theta_u^{\text{AF}} \le \min\Big\{\frac{\kappa_u}{k},\ \frac{\kappa_u}{\kappa_u+(1-\phi\rho_u)(1-\epsilon)(\sigma+m\frac{T_{\text{lag}}-1}{2T_{\text{lag}}}q)}\Big\}. \tag{4}$$

These reduce to baseline bounds when ϕ=0 and give the ideal-collusion envelope when ϕ=1.

Repeated observations with collusion. For s_u observations from the same sender, Eq. (5) extends to

$$\Pr\big[\exists \text{ owner in } s_u \text{ draws}\big] \le s_u \min\big\{\tfrac{\kappa_u}{k},\ \tfrac{\kappa_u}{\kappa_u+(1-\phi\rho_u)X_u}\big\}. \tag{5}$$

Collusion increases the effective observation count and reduces obfuscating mass, but the cap κ_u/k continues to govern attribution success for honest senders (the bound is a union bound and may be loose for large s_u).

C. Attack Strategies and Metrics

Why we focus on attribution attacks. Standard defenses against membership inference or reconstruction attacks [6]–[8] are orthogonal and can be layered atop FLTorrent. Our focus is solely on *within-round attribution* introduced by BitTorrent-style dissemination, i.e., inferring whether an observed sender is also the true source of the transmitted chunk/update.

Metric: Attribution Success Rate (ASR). For a receiver v, consider all warm-up transfers it observes in a round. An attacker outputs, for each observed transfer, a binary attribution decision (or a ranked guess) for whether $\mathsf{src}(c) = \mathsf{snd}(c)$. We report ASR as the fraction of observed transfers that are attributed correctly; in evaluation we often use the *maximum* ASR over receivers (and over coalition members) as a conservative summary. This metric is *empirically* related to the bounds in Section IV-A: Eq. (1) caps per-transfer attribution for honest senders, while Eq. (5) captures how aggregating evidence across multiple transfers can increase success.

Three observation-only attack strategies. All attacks below are *observation-only* and fit Adversary A in Section II-C (honest-but-curious, possibly colluding). They do not inspect payload contents or manipulate protocol state.

(1) **Sequential Greedy Strategy.** The attacker labels the *first* chunk received from each sender pseudonym as its owner, targeting the strongest early-round signal and instantiating the "early owner bias" that warm-up aims to suppress.

(2) **Amount Greedy Strategy.** For each observed sender pseudonym, the attacker aggregates evidence across transfers by grouping received pieces by their update/torrent descriptor (the adversary does not know the producing client's real identity, only descriptor identities). It then attributes the sender to the descriptor that appears most frequently among the sender's early transfers. Intuitively, if a sender transmits disproportionately many chunks from its own (unknown-to-peers) update early, frequency becomes a strong attribution cue.

(3) **Clustering Strategy.** The attacker forms a feature vector per sender pseudonym using (i) temporal features (arrival order statistics) and (ii) frequency features (counts per observed descriptor/chunk family), then clusters sender pseudonyms against descriptor-level IDs and outputs the best match. This combines the early-time and volume signals and captures more sophisticated inference without requiring payload inspection.

Relation to our bounds. Sequential Greedy approximates the strongest single-shot posterior, while Amount Greedy and Clustering exploit multiple observations (s_u) from a sender. Equations (1) and (5) therefore upper-bound attacker advantage as k grows (and tighten further with mixing in Eq. (2)), matching the empirical ASR trends reported in our evaluation.

V. Performance Evaluation

This section evaluates *FLTorrent* along four axes: (i) **learning utility** (accuracy vs. CFL and GossipDFL), (ii) **warm-up efficiency** (bandwidth utilization vs. a max-flow upper bound and heuristics), (iii) **end-to-end round cost** (warm-up share and scalability), and (iv) **privacy efficacy** under *within-round source unlinkability* objective (**attack success rate**, ASR).

Semantics and scope of evaluation. FLTorrent follows the FedAvg update rule and differs from CFL only in the *dissemination substrate*. As in Section II-B, each client aggregates over the *reconstructable active set* by the round deadline. In our learning experiments, we set the deadline so updates are reconstructable with high probability; otherwise, we fall back to standard partial-participation semantics (FedAvg over the reconstructable active set) under stragglers or disconnects.

Metrics. We report: (1) **Bandwidth utilization** during warm-up; (2) **Warm-up duration** T_{warm} (until *active* clients reach the warm-up threshold and switch to vanilla BitTorrent); (3) **Round duration** T_{round} (warm-up + BitTorrent dissemination; local aggregation is executed after dissemination); (4) **ASR**: the fraction of correct attributions under the attack models in Section IV-C and threat model in Section II-C.

TABLE II: Test accuracy (mean±std) on CIFAR-10 and MNIST (n=50 clients, local epochs = 5, batch size = 32, 50 communication rounds). FLTorrent consistently surpasses GossipDFL and remains close to CFL.

	Distribution	CFL	GossipDFL	FLTorrent
CIFAR-10	Dir(α=0.1)	0.624±0.025	0.379±0.031	0.621±0.027
	Dir(α=0.5)	0.667±0.023	0.633±0.027	0.671±0.024
	Dir(α=1.0)	0.724±0.023	0.709±0.025	0.722±0.022
	IID	0.793±0.017	0.734±0.021	0.791±0.019
MNIST	Dir(α=0.1)	0.979±0.006	0.907±0.007	0.978±0.007
	Dir(α=0.5)	0.987±0.005	0.965±0.004	0.988±0.004
	Dir(α=1.0)	0.989±0.003	0.972±0.004	0.990±0.003
	IID	0.992±0.002	0.986±0.003	0.992±0.002

A. Setting

Model and chunking. Unless otherwise stated, we slice a *GoogLeNet* update into 206 chunks of 256 KiB each ($206 \times 256\,\text{KiB} = 51.5\,\text{MiB} \approx 54.0\,\text{MB}$ in decimal units, consistent with the $\sim$51–54 MB scale reported in [25]).

Access links. We sample heterogeneous uplink/downlink capacities based on European residential broadband statistics [26]: uplink 15.5–25.3 Mbps and downlink 36.5–121 Mbps, corresponding to roughly $[7, 12]$ and $[18, 60]$ chunks/s respectively for 256 KiB chunks.

Overlay. We use a random overlay with minimum degree m (default m=10) and heterogeneous neighbor counts above m.

Warm-up knobs (unlinkability hardening). We evaluate the three warm-up mechanisms from Section III-B: `K` (cover-set gating / warm-up threshold), `PR` (pre-round obfuscation), and `TL` (time obfuscation). By default, we use `PR` ratio R=0.2 (i.e., $\lfloor 0.2 \times 206 \rfloor = 41$ chunks per update) and time-lags $\ell_v \sim \text{Unif}\{0, 1, 2\}$ seconds (slotted time with $\Delta = 1$ s). When we sweep the warm-up threshold, we report it as `K` in percentage of the *swarm-wide* chunk universe (Figure 5); the default is `K`= 10%. Equivalently, in the analysis notation (Section II-D) this corresponds to $k = \lceil \texttt{K} \cdot |\mathcal{C}^r| \rceil$ where the global chunk universe $|\mathcal{C}^r| = \sum_{v \in \mathcal{V}} K_v^r$ (here $|\mathcal{C}^r| = 206n$ for n clients each contributing 206 chunks). For example, at n=500 and `K`= 10%, we have $|\mathcal{C}^r|$=103,000 and k=10,300 pieces ($\approx$ 2.5 GiB per client at 256 KiB/piece). We use `K` (a fraction of $|\mathcal{C}^r|$) so that the cover-set knob scales with swarm size and yields comparable unlinkability across n.

B. FLTorrent vs. Gossip DFL vs. CFL

To assess convergence and final accuracy, we compare *FLTorrent* with centralized federated learning (CFL, a pragmatic upper bound) and a representative mix-and-forward decentralized method, *Gossip DFL* [27], [28]. We evaluate on MNIST and CIFAR-10 under IID and Non-IID client data. Non-IID partitions follow a Dirichlet sampler with $\alpha \in \{0.1, 0.5, 1.0\}$ (smaller α implies stronger heterogeneity). Each experiment runs for 50 communication rounds. For readability, Figure 2 plots representative non-IID cases ($\alpha \in \{0.1, 0.5\}$) plus IID; Table II additionally reports $\alpha = 1.0$. Unless otherwise stated, we report mean test accuracy across 10 seeds, with variability summarized as mean±std. All methods use identical hyperparameters (optimizer, learning rate, etc.) on each dataset to ensure fair comparison.

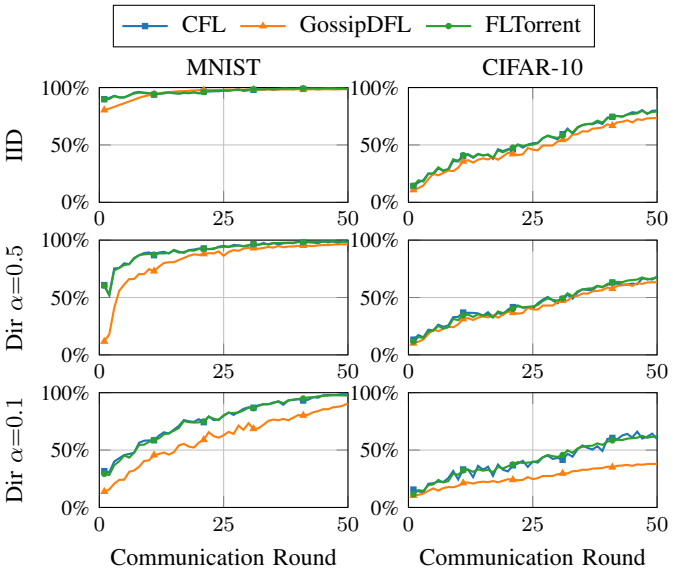

Fig. 2: Convergence on MNIST and CIFAR-10 under IID and Dirichlet non-IID splits ($\alpha \in \{0.5, 0.1\}$). Test accuracy vs. communication rounds for CFL, GossipDFL, and FLTorrent.

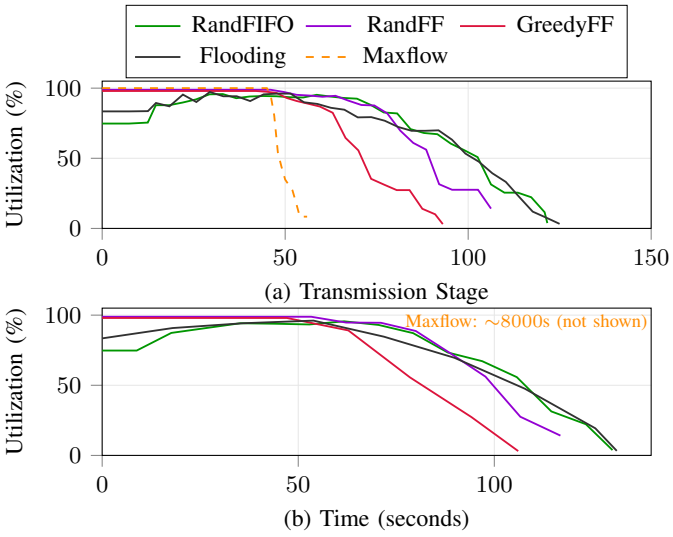

Fig. 3: Warm-up bandwidth utilization for online heuristics vs. a max-flow upper bound.

Overall, FLTorrent matches (closely tracks) CFL and consistently outperforms GossipDFL, especially under strong heterogeneity where local mixing attenuates global information.

C. End-to-End Round Communication Cost

We next quantify the communication overhead of unlinkability hardening, focusing on (i) warm-up bandwidth efficiency and (ii) end-to-end round time decomposition. Figure 3 shows that warm-up sustains high utilization across heuristics. **GreedyFastestFirst** completes warm-up earlier while approaching the max-flow upper bound in the high-utilization regime; unless otherwise stated, we use it below.

Control plane. Bitfields and stage schedules are only a few KiB per client and are negligible relative to the multi-GiB

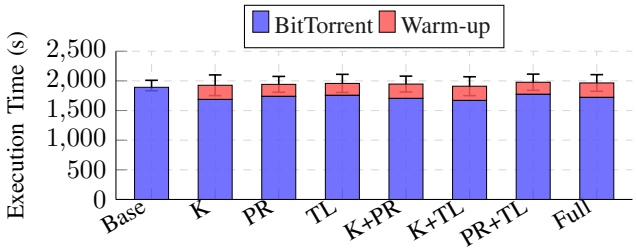

Fig. 4: End-to-end time decomposition under privacy ablations (K, PR, TL), splitted into the BitTorrent and the warm-up.

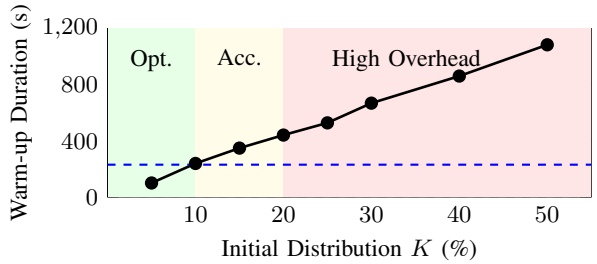

Fig. 5: Warm-up duration as the warm-up threshold K increases (expressed as % of the swarm-wide chunk universe).

data plane; tracker integrity is discussed in §III-D. Figure 4(a) decomposes the round time (100 nodes). With all defenses enabled (Full), warm-up takes 243.32 s and the subsequent BitTorrent phase takes 1721.75 s (total 1965.07 s). Compared to BitTorrent-only (Base, 1891.75 s), the *total* overhead is modest ($\approx$ 3.9%), because warm-up improves early chunk diversity and shortens the later BitTorrent phase. Within Full, adding PR and TL on top of K has small marginal cost (warm-up 238.78 s $\rightarrow$ 243.32 s). Figure 5 quantifies the deployment-time privacy–cost knob K. Warm-up ends once every *active* client reaches the minimum cover set (§III-B); in evaluation we report this as K, which corresponds to the k-chunk threshold in our analysis. As K increases, warm-up grows monotonically: $\approx$ 99.5 s at K= 5%, $\approx$ 238.8 s at K= 10%, and $\approx$ 1084.7 s at K= 50%. Thus, moderate thresholds provide substantial early-round mixing at manageable cost, whereas aggressive thresholds can dominate end-to-end time.

Table III shows stable scaling from 100 to 500 peers: warm-up remains a consistent $\approx$ 11.5%–12.4% share of round time and utilization stays in the 75%–80% range.

D. Privacy Evaluation: Within-round Source Unlinkability

We now quantify privacy under the threat model in Section II-C using the attacks in Section IV-C. Unless otherwise stated, we use a 100-node overlay with minimum degree m=10. As a coarse reference point, if an adversary cannot do better than *uniform guessing among m candidate neighbors* for first-hop attribution, its success rate is $\approx 1/m$ (10% at m=10). Our goal is to push empirical ASR toward this regime.

Connecting ASR to the analysis. Section IV-A caps the *per-transfer* attribution posterior for honest senders (Eq. (1)) and characterizes further tightening due to warm-up mixing (Eq. (2)). ASR, however, is computed over *multiple* observations and can aggregate evidence across several transfers from

TABLE III: End-to-end round cost under Full privacy (GreedyFastestFirst, 51 MB model, chunk=256 KiB).

n	T_{warm} (s)	Share (%)	Util. (%)	T_{round} (s)
100	243	12.4	80.5	1965
200	472	11.5	76.3	4118
300	730	11.7	77.1	6229
400	939	11.5	79.4	8192
500	1229	11.7	75.4	10501

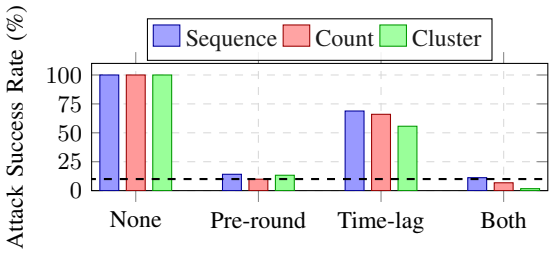

Fig. 6: Privacy ablation: maximum ASR under three inference strategies (Sequence, Count, Cluster).

the same sender (cf. Eq. (5)), which explains why empirical ASR need not numerically equal the single-transfer cap. Figure 6 shows that unlinkability hardening is necessary: without defenses ASR is near-perfect, pre-round obfuscation provides the dominant reduction, time-lags alone are insufficient, and combining them pushes ASR close to the neighborhood-level random-guess baseline. Figure 7 studies parameter sensitivity under Full defenses. Increasing overlay density is the most effective structural lever: as m increases from 5 to 25, max ASR drops from 26.99% to 4.29%. To provide a moving reference in the m sweep, Figure 7(a) also plots the random-guess baseline for *max* ASR, $\approx 1/m$ (since we enforce a minimum degree of m). Pre-round volume R exhibits diminishing returns: varying R from 10% to 50% changes max ASR only slightly (11.43% to 11.27% in our sweep). Scaling the network provides additional privacy amplification: as n increases from 100 to 500, Sequence ASR *decreases* from 10.90% to 7.31%, and Count ASR increases from 5.58% to 6.78% (still below the $1/m$ level). Notably, with $m = 10$ fixed, the three attack strategies yield ASR close to (and often below) the random-guess baseline across all network sizes, confirming that the defense remains effective at scale. This trend aligns with intuition: larger networks provide more opportunities for chunk relay, increasing the non-owner mass in each sender's cover set and thereby reducing attribution confidence. Finally, coalition analysis shows that while the probability that *at least one* attacker succeeds grows with coalition size a (13.56% at a=5 to 30.82% at a=25), the *per-attacker* ASR remains comparatively low (11.31% to 14.32%), indicating resilience against straightforward collusion under our observation-only threat model.

E. LLM-Scale Round-Time Overhead

We use LLM-scale runs as *systems stress tests* for dissemination: they validate end-to-end *relative* overheads under very large artifacts, without requiring the (much costlier) end-to-end fine-tuning convergence evaluation. Accordingly, we

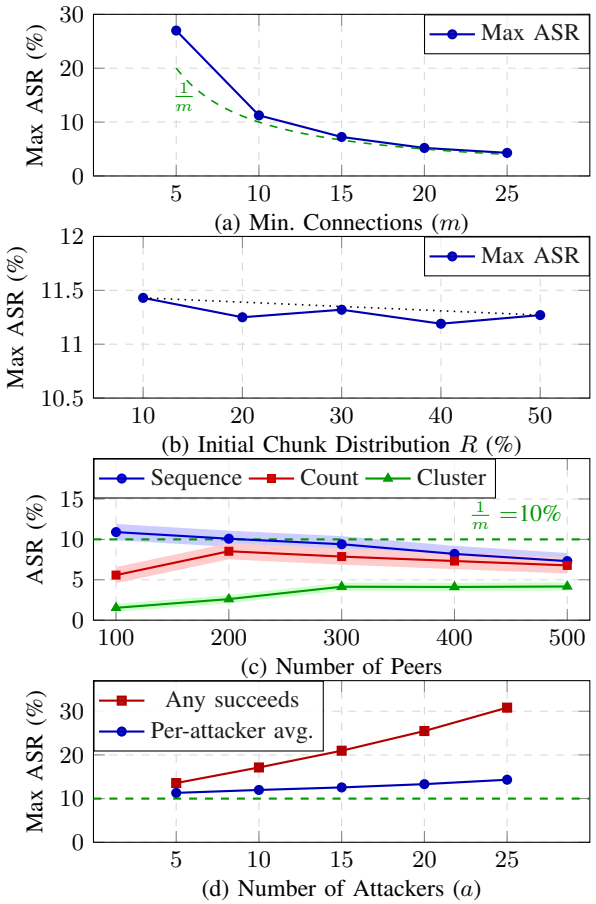

Fig. 7: ASR under warm-up defenses.

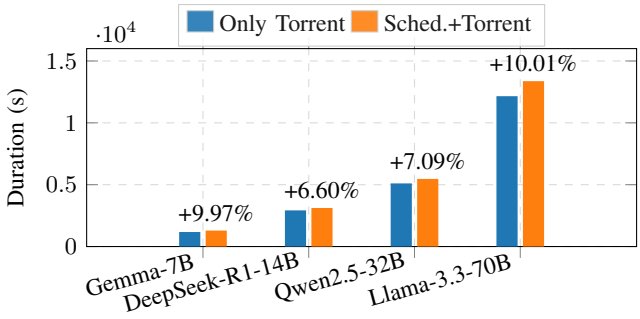

Fig. 8: Round duration overhead of FLTorrent (with unlinkability hardening) vs. BitTorrent-only for LLM-scale updates.

do *not* use the LLM setting to argue learning convergence; learning utility is evaluated above on MNIST/CIFAR-10 under residential broadband conditions (15–25 Mbps uplink). LLM-scale experiments use datacenter-class links (7–10 Gbps) to stress-test dissemination at artifact sizes where residential links would be impractically slow. The unlinkability mechanisms (warm-up, pre-round obfuscation, scheduling) are identical in both settings; we therefore interpret Figure 8 strictly as dissemination overhead (%), not as an end-to-end learning claim. Figure 8 compares FLTorrent against BitTorrent-only for four LLMs (Gemma-7B [29], DeepSeek-R1-14B [30], Qwen2.5-32B [31], and Llama-3.3-70B [32]) over 7–10 Gbps access links. FLTorrent adds a modest end-to-end overhead (9.97%, 6.60%, 7.09%, and 10.01% respectively), while providing the unlinkability gains quantified in Figures 6–7.

VI. Related Work

Decentralized learning and communication efficiency. Decentralized learning (DL) replaces a central aggregator with peer-to-peer mixing over an overlay graph. Representative lines include compression-aware decentralized optimization (e.g., CHOCO-style) and unified convergence analyses [14], [28], as well as relay- and topology-accelerated designs to improve mixing speed [33], [34]. Systems work further studies deployment factors such as bandwidth heterogeneity and communication overhead [35], [36]. These efforts optimize learning and communication, but typically do not treat the *dissemination layer* as a first-class privacy surface.

BitTorrent and P2P dissemination. BitTorrent disseminates objects by chunking and parallel swarming and achieves high throughput under heterogeneity and churn [17], [37]; it has also been used to exchange learning artifacts [38]. However, vanilla BitTorrent does not target *source unlinkability*: early in a swarm, peers predominantly hold their own chunks, making early transfers highly attributable, and BitTorrent metadata can enable identification/profiling [39]. Privacy-enhanced substrates (e.g., onion-routing/mix-like overlays or friend-to-friend swarms) reduce linkability but typically incur substantial throughput and/or deployment complexity. In contrast, FLTorrent formulates *within-round source attribution* as a privacy risk for BitTorrent-style FL dissemination and mitigates it via a short warm-up that establishes a minimum cover set *before* reverting to standard swarming.

Privacy leakage and attribution risks in FL. FL privacy work largely focuses on *content leakage* (e.g., membership inference and reconstruction) [6]–[8] and defenses such as secure aggregation and differential privacy [10]. These protections are complementary: even with content protection, dissemination can leak *attribution metadata* linking updates to participants, consistent with *source inference* showing such linkage can be actionable [19], [20]. FLTorrent targets *protocol-level within-round attribution* induced by swarming metadata and can be combined with secure aggregation/DP when both metadata-level unlinkability and content-level confidentiality are desired. Cross-round linkability is important, but our formal guarantee focuses on the within-round surface.

Anonymity and unlinkability for FL communication. Several proposals provide anonymous participation or unlinkable submission/authentication via cryptographic mechanisms or redesigned communication substrates [40], [41]. These often target stronger anonymity notions and require heavier protocol changes. FLTorrent is orthogonal: it preserves BitTorrent-grade efficiency while addressing the attribution channel that arises *even when participants are known* (e.g., permissioned cross-silo consortia) with a short warm-up prior to vanilla swarming.

Positioning. DL work optimizes convergence/efficiency and most FL privacy defenses protect update *contents*; BitTorrent-style dissemination adds a protocol-level attribution channel via early owner bias and proximity-visible metadata. FLTorrent

fills this gap with a privacy-preserving warm-up (cover-set threshold, owner throttling, and non-owner-first scheduling) and then reverts to swarming to retain scalability in large, heterogeneous deployments.

VII. Conclusions

We presented *FLTorrent*, a BitTorrent-based dissemination layer for decentralized federated learning that mitigates a *protocol-level* privacy risk—*within-round source attribution*—while retaining BitTorrent-grade swarming efficiency. Since vanilla BitTorrent is not designed for *source unlinkability*, early transfers can make "who sent what" strongly attributable from early owner bias and proximity-visible metadata. FLTorrent introduces a short warm-up that enforces a minimum cover set (cover-set k-threshold, owner throttling, and non-owner-first scheduling) before reverting to vanilla swarming. We bounded per-transfer attribution confidence by a $1/k$-type term (more generally K_u/k), with extensions to colluding adversaries and a *malicious-but-detectable* tracker model. Empirically, our greedy warm-up scheduler reaches $\sim$92% of a max-flow upper bound; warm-up remains $\sim$12% of round time with 75%–80% utilization over 100–500 peers; and LLM-scale stress tests over 7–10 Gbps links show only modest overhead versus BitTorrent-only while reducing ASR to *below* random guessing under our observation-only threat model—with privacy amplification as network size increases. FLTorrent is complementary to content-protection mechanisms (secure aggregation, differential privacy). Future work includes robustness to stronger network/active adversaries, Sybil resistance for open membership, and decentralizing the tracker while preserving auditability and unlinkability.

References

[1] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *AISTATS*, 2017.

[2] B. Luo, Y. Feng, S. Wang, J. Huang, and L. Tassiulas, "Incentive mechanism design for unbiased federated learning with randomized client participation," in *ICDCS*, 2023.

[3] X. Wu, Z. Ji, and C.-L. Wang, "Embedding communication for federated graph neural networks with privacy guarantees," in *ICDCS*, 2023.

[4] H. Zhang, Y. Xie, S. Hu, M. He, P. He, J. Zheng, and D. Feng, "Fedlth: A privacy-preserving federated learning framework with model pruning on edge clients," in *ICDCS*, 2025.

[5] J. Xia, W. Wu, L. Luo, G. Cheng, D. Guo, and Q. Nian, "Accelerating and securing federated learning with stateless in-network aggregation at the edge," in *ICDCS*, 2024.

[6] R. Shokri, M. Stronati, C. Song, and V. Shmatikov, "Membership inference attacks against machine learning models," in *IEEE S&P*, 2017.

[7] L. Zhu, Z. Liu, and S. Han, "Deep leakage from gradients," *NeurIPS*, 2019.

[8] J. Geiping, H. Bauermeister, H. Dröge, and M. Moeller, "Inverting gradients-how easy is it to break privacy in federated learning?," *NeurIPS*, 2020.

[9] F. Diana, A. Nusser, C. Xu, and G. Neglia, "Cutting through privacy: A hyperplane-based data reconstruction attack in federated learning," 2025. arXiv:2505.10264.

[10] K. Bonawitz, V. Ivanov, B. Kreuter, A. Marcedone, H. B. McMahan, S. Patel, D. Ramage, A. Segal, and K. Seth, "Practical secure aggregation for privacy-preserving machine learning," in *ACM CCS*, 2017.

[11] D. Pasquini, D. Francati, and G. Ateniese, "Eluding secure aggregation in federated learning via model inconsistency," in *ACM CCS*, 2022.

[12] K.-H. Ngo, J. Östman, G. Durisi, and A. Graell i Amat, "Secure aggregation is not private against membership inference attacks," 2024. arXiv:2403.17775.

[13] M. Assran, N. Loizou, N. Ballas, and M. Rabbat, "Stochastic gradient push for distributed deep learning," in *ICML*, 2019.

[14] A. Koloskova, N. Loizou, S. Boreiri, M. Jaggi, and S. Stich, "A unified theory of decentralized sgd with changing topology and local updates," in *ICML*, 2020.

[15] L. Yuan, Z. Wang, L. Sun, P. S. Yu, and C. G. Brinton, "Decentralized federated learning: A survey and perspective," *IEEE Internet Things J.*, vol. 11, no. 21, 2024.

[16] D. Pasquini, M. Raynal, and C. Troncoso, "On the (in)security of peer-to-peer decentralized machine learning," in *IEEE S&P*, 2023.

[17] B. Cohen, "Incentives build robustness in bittorrent," in *Workshop on Economics of P2P Systems*, 2003.

[18] R. Cuevas, N. Laoutaris, X. Yang, G. Siganos, and P. Rodriguez, "Deep diving into bittorrent locality," *ACM SIGMETRICS Perform. Eval. Rev.*, vol. 38, no. 1, 2010.

[19] H. Hu, Z. Salcic, L. Sun, G. Dobbie, and X. Zhang, "Source inference attacks in federated learning," in *ICDM*, 2021.

[20] H. Chen, X. Xu, X. Zhu, X. Zhou, F. Dai, Y. Gao, X. Chen, S. Wang, and H. Hu, "Where does this data come from? enhanced source inference attacks in federated learning," in *IJCAI*, 2025.

[21] B. Pejó and G. Biczók, "Quality inference in federated learning with secure aggregation," *IEEE Trans. Big Data*, vol. 9, no. 5, 2023.

[22] BitTorrent Enhancement Proposals, "Bep 0003: The bittorrent protocol specification." https://www.bittorrent.org/beps/bep_0003.html, 2003.

[23] J. D. Ullman, "Np-complete scheduling problems," *J. Comput. Syst. Sci.*, vol. 10, no. 3, 1975.

[24] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W.H. Freeman, 1979.

[25] G. Shu, W. Liu, X. Zheng, and J. Li, "If-cnn: Image-aware inference framework for cnn with the collaboration of mobile devices and cloud," *IEEE Access*, vol. 6, 2018.

[26] "Oecd." https://www.oecd.org, 2024.

[27] X. Lian, C. Zhang, H. Zhang, C.-J. Hsieh, W. Zhang, and J. Liu, "Can decentralized algorithms outperform centralized algorithms? a case study for decentralized parallel stochastic gradient descent," *NeurIPS*, 2017.

[28] A. Koloskova, S. Stich, and M. Jaggi, "Decentralized stochastic optimization and gossip algorithms with compressed communication," in *ICML*, 2019.

[29] "Gemma open models." https://blog.google/technology/developers/gemma-open-models/, 2024.

[30] D. Guo, D. Yang, H. Zhang, *et al.*, "Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning," *arXiv:2501.12948*, 2025.

[31] Qwen Team, "Qwen2 technical report," *arXiv:2407.10671*, 2024.

[32] A. Dubey, A. Jauhri, A. Pandey, *et al.*, "The llama 3 herd of models," *arXiv:2407.21783*, 2024.

[33] T. Vogels, L. He, A. Koloskova, S. P. Karimireddy, T. Lin, S. U. Stich, and M. Jaggi, "Relaysum for decentralized deep learning on heterogeneous data," in *NeurIPS*, 2021.

[34] B. Ying, K. Yuan, Y. Chen, H. Hu, P. Pan, and W. Yin, "Exponential graph is provably efficient for decentralized deep training," *NeurIPS*, 2021.

[35] A. Dhasade, A.-M. Kermarrec, R. Pires, R. Sharma, M. Vujasinovic, and J. Wigger, "Get more for less in decentralized learning systems," in *ICDCS*, 2023.

[36] C. Chen, M. Li, and C. Yang, "bbtopk: Bandwidth-aware sparse allreduce with blocked sparsification for efficient distributed training," in *ICDCS*, 2023.

[37] A. Legout, G. Urvoy-Keller, and P. Michiardi, "Rarest first and choke algorithms are enough," in *ACM IMC*, 2006.

[38] O. Carl and T. Weis, "Bittorrent-based gossip learning," in *ACM IoT*, 2024.

[39] S. Le Blond, A. Legout, F. Lefessant, W. Dabbous, and M. A. Kaafar, "Spying the world from your laptop: identifying and profiling content providers and big downloaders in bittorrent," in *USENIX LEET*, 2010.

[40] G. Almashaqbeh and Z. Ghodsi, "Anofel: Supporting anonymity for privacy-preserving federated learning," *PoPETs*, 2025.

[41] A. Agiollo, E. Bardhi, M. Conti, N. Dal Fabbro, and R. Lazzeretti, "Anonymous federated learning via named-data networking," *Future Gener. Comput. Syst.*, vol. 152, 2024.