THOX.ai Research · Technical report v1.1.0 · 2026-11-03

Preprint, not peer reviewed

THOX MeshStack: Capacity-Priced Pipeline Sharding, Grant-Gated KV Transfer and Lease-Fenced Device Roles for a Private Household Mesh

Two design lineages reconciled into one specification, with the parts that run on a general host measured, the parts that depend on public device floors calculated, and the parts that need a fleet stated as a gated protocol. Edition v1.1.0 (evidence edition) adds Virtual Lab measurements on fleet host classes and corrects the codec ratio, the 27B state geometry and the transfer overhead against them.

THOX.ai Research · Authors: Craig Ross, Chief Executive Officer, and Tommy Xaypanya, CTO and Co-Founder (corresponding author), THOX.ai LLC, Cedar Park, Texas · ORCID 0009-0000-4425-9565

ABSTRACT

A household owns several small computers that are idle most of the day and together hold more memory than any one of them. Today every serving path in the THOX fleet routes a whole request to one node that already holds a whole model, so the largest model a home can run is bounded by its largest device. This paper reconciles two THOX design lineages for MeshStack™, the private device mesh, into one specification and tests the parts of it that can be tested without a device fleet. The design has four mechanisms: a capacity-priced pipeline sharder whose two-domain admission rule fails closed; a KV cache organised as a parent-chained, HMAC-keyed graph whose cross-node transfer is gated by both a cost model and a live authority path; a lease-fenced role layer that keeps device owners sovereign under routine churn; and a control plane that degrades to a 239-byte LoRa carrier when the LAN is gone. We report measured results for the mechanisms that run on general hosts: lossless compression of real KV state captured from running GGUF inference reaches 1.17x to 1.20x with a byte-split zstd codec, bit-exact, at 422 to 935 MB/s encode on a CPU host class (the 1.51x of the previous edition was a bfloat16-tensor result and is restated in Section 6.1); parent-chained HMAC keying costs 9.7 microseconds per 64-token chunk; a cross-host KV transfer and a two-host llama.cpp pipeline split were measured on fleet host classes with their caveats stated; and the lease fencing of Section 4.5 passed a live drill on the deployed coordinator, with every stale action rejected. We report calculated results where the inputs are public: at the published THOX class floors (2, 4 and 6 GB) a three-device mesh cannot place the 27.8B demo model used by the current reference implementation at any usable fraction we tried, which contradicts the reference demo's fixtures and is the paper's central negative result. No THOX device measurement is claimed; every measured number is labelled with the host class it ran on.

CONTENTS

Executive summary
1. Introduction
2. Background and related work
3. System and threat model
4. Design
5. Implementation
6. Evaluation
7. Discussion and limitations
8. Conclusion
9. Roadmap and decisions for the founders
Appendix A. Reproducibility
Appendix B. Notation

Executive summary

EDITION NOTE: V1.1.0 EVIDENCE EDITION

This edition corrects v1.0.1 against Virtual Lab measurements on fleet host classes. (1) The KV compression headline is now the real-GGUF result, 1.17x to 1.20x; v1.0.1's 1.51x is restated as a bfloat16-tensor result (Section 6.1). (2) The 27B admission geometry is corrected to the measured 65,536 B/token of attention KV plus a fixed 156,894,356 B recurrent state per sequence; the admission outputs are recomputed with the paper's own planner and do not change (Section 6.3). (3) The PLACE per-transfer overhead is the measured 0.196 s, not the assumed 0.02 s, and the crossover analysis is recomputed on measured inputs (Section 6.4). (4) The fencing mechanism is described as deployed, and a live drill on the production coordinator is reported (Section 6.6). (5) A two-host engine split is measured for the first time, at n = 1 with its transport caveats (Section 6.7). The design, the negative capacity result and the protocol propositions are unchanged. Citations use the Zenodo concept DOI 10.5281/zenodo.23919579; the v1.1.0 version DOI is pending deposit.

MeshStack™ exists to spend a household's aggregate device memory on models no single device can hold. Two THOX design documents answer how: the storage-aware parallelism paper of edition 1.1.0 [1], which recommends whole-request placement first and pipeline distribution only after measurement, and the MeshStack v2 design paper [2], which fixes pipeline sharding as the mesh form and adds a KV graph, a secrets graph and a scheduler cost form. A third note proposes expert parallelism and speculative decoding on top [3], and a fourth proposes a lease-based device role layer [4]. They agree on principles and disagree on sequencing, control-plane ownership and what the device floors permit. This paper is the reconciliation, and it adds the evidence that could be produced without a fleet.

1.17 to 1.20×
lossless compression of real KV captured from GGUF inference (f16), byte-split zstd, bit-exact, Azure VM CPU host class; v1.0.1's 1.51x was bfloat16 tensors
MEASURED

9.7 µs
per 64-token parent-chained HMAC-SHA256 chunk key; 4096-token chain 0.37 ms
MEASURED

Unplaceable
27.8B Q4 demo model on an Air+Mini+Clip trio at the public floors (12 GB total vs 14.6 GiB weights); unchanged under the corrected state geometry of Section 6.3
CALCULATED

0
stale authorisations accepted in the live fencing drill on the deployed coordinator (lease re-granted 15 to 16 after the holder departed; every stale action rejected). Simulation: 3 / 3 stale completions rejected per churn run
MEASURED (LIVE DRILL)

PRODUCT IMPLICATIONS

First, the demo fixtures in the current reference implementation (a 9.6 GiB usable anchor) describe a host, not a THOX device; at the published class floors the trio needs a hub for anything in the 27B class, so the first shipped MeshStack capability must be stated as device-plus-hub, not device-only. Second, lossless wire compression of real GGUF KV is worth 1.17x to 1.20x (f16, bit-exact, CPU host class). The 1.51x of the previous edition was measured on bfloat16 tensors; casting the same captured tensors to bfloat16 recovers 1.42x to 1.46x with the same codec, so most of that difference is the tensor precision, not the codec. The PLACE cost model should use a measured ratio per model, never a constant, and the lossless-transport note's expectation of large gains is not supported. Third, the role and lease layer proposed outside the v2 paper fits the v2 secrets graph without a second control plane: a lease is a grant with an epoch and a TTL. Fourth, a LoRa auxiliary mesh can carry presence, heartbeats and revocation for a household-sized mesh, and nothing else; it is a control-plane fallback, not a transport.

1 Introduction

The THOX device line (ThoxAir, ThoxMini, ThoxClip, the ThoxKey companion and desktop hosts) runs ThoxOS™, a local-first operating system whose public class floors are 2 GB, 4 GB and 6 GB of RAM for the E2, P4 and P6 classes [5]. A fleet audit of 34 MeshStack-related repositories on 2026-10-30 found that every serving path routes a whole request to a node that already holds the whole model [2]. That pattern is correct for models that fit and useless for models that do not. The design question is how to pool the memory of a few small, churn-prone, owner-controlled devices without pretending the home is a datacenter.

Published systems answer the capacity half of that question. prima.cpp runs 30B to 70B models on heterogeneous home clusters with pipelined-ring parallelism and

disk offload [6]; TPI-LLM argues for tensor parallelism on low-resource edge devices [7]; EdgeShard, Galaxy and LinguaLinked partition models across cooperating edge devices [8, 9, 10]; Petals and BloomBee run pipelines over the internet with checkpoint-and-replay recovery and lossless activation compression [11, 12, 13]; exo is the best-known hobbyist implementation [14]. None of them answers the authority half: who may move a household's KV state, when a departed device's permissions die, and how an owner's foreground work stays ahead of mesh jobs. That half is where THOX's design differs, and it is where this paper spends its design effort.

Within THOX the record is split. Lineage A [1] is a source-grounded architecture over five parallelism families with a calculator and an unexecuted physical protocol; it recommends whole-request placement and local staging first. Lineage B [2] fixes pipeline sharding, a five-tier KV graph with grant-gated transfer, a secrets grant graph and a Dynamo-style cost form, with a Python reference implementation whose root suite passed 63 tests at HEAD b01bc03 after a recorded admission regression. Two notes add a lossless-transport proposal with expert parallelism [3] and a lease-based Auto Device Role Mode [4]. The contributions of this paper are:

1. **A reconciled specification** (Section 4) that keeps lineage B's mechanisms, adopts lineage A's sequencing (whole-request placement is the degenerate one-stage case of the sharder and ships first), folds the lease and role proposal into the secrets graph as epoch-bearing grants, and rejects expert parallelism and speculative decoding for generation 1 with stated reasons.
2. **Measured lossless wire behaviour** (Sections 6.1, 6.2): on real KV captured from running GGUF inference the byte-split zstd codec reaches 1.17x to 1.20x (f16, bit-exact, CPU host class), and parent-chained HMAC keying costs 9.7 microseconds per 64-token chunk. The bfloat16-tensor numbers of the previous edition are restated as harness results in Section 6.1. These numbers bound the PLACE cost model and the KV graph's overhead on general host classes.
3. **A negative capacity result** (Section 6.3): at the public class floors, with first-fit contiguous placement and stated usable fractions, the reference demo's 27.8B Q4 geometry is Unplaceable on an Air+Mini+Clip trio and needs either a hub or a larger number of Clip-class nodes (6 at a 50% usable fraction and 2048-token context).
4. **Protocol evidence by simulation and by live drill** (Section 6.6): epoch fencing rejects every stale completion and consumption-time validity removes the stale authorisation window under stated churn distributions, and the deployed coordinator passed a live departure drill with zero stale authorisations accepted.
5. **Cross-host evidence** (Sections 6.4, 6.7): a grant-gated KV transfer measured between two fleet host classes, with its per-transfer overhead measured rather than assumed, and a two-host llama.cpp pipeline split measured at n = 1 with

the transport caveats stated. Generation-phase wire bytes per token were not isolated, and no 27B split was achieved; both are stated plainly.

6. **A control-plane degradation analysis** (Section 6.5): the LoRa airtime of each MeshStack control message on every Meshtastic preset, and the number of nodes whose lease heartbeats fit a regulatory duty cycle.

Everything the paper measures ran on general fleet host classes (an Azure VM CPU host, a fleet GPU host, an EC2 Mac with an Apple M4, and the deployed role coordinator), each labelled where its numbers appear. Nothing ran on a THOX device, and no device measurement is claimed anywhere in this paper. Section 7 states those limits once more, itemised, and Section 9 lists the decisions that only the founders can take.

2 Background and related work

2.1 Parallelism forms and why the mesh form is pipeline

Tensor parallelism slices every layer's matrices across workers and needs a collective per layer, which assumes a fast symmetric interconnect [15]. Pipeline parallelism assigns contiguous layer ranges to stages and moves one activation tensor per token or micro-batch per boundary. Expert parallelism places experts of a mixture-of-experts layer on different workers and routes tokens to them [16]. Context parallelism partitions sequence or attention state [17]. Lineage A translated all five families into a resource contract and found that only whole-request and pipeline placement are admissible over slow links without backend support that does not yet exist in the fleet [1]; lineage B fixed pipeline as the mesh form and tensor as intra-node only (decision D10) [2]. TPI-LLM's counter-argument, that tensor parallelism can beat pipeline for single-user edge inference, holds when links are fast enough for per-layer collectives and memory is the binding constraint [7]; its evaluation used a controlled testbed, and the THOX relay and Wi-Fi assumptions of Section 3 do not satisfy its premise. This paper keeps pipeline as the generation-1 form and records tensor-on-LAN as a measured question for the benchmark wave.

2.2 KV cache as a movable resource

Datacenter serving systems treat the KV cache as a first-class, tiered, movable object: Mooncake builds a KV-centric disaggregated architecture [18], LMCache a KV layer across engines and tiers [19], CacheBlend reuses precomputed KV at non-prefix positions by selective recomputation [20], CacheGen streams compressed KV over bandwidth-limited links [21], and PagedAttention manages KV inside one engine [22]. Orca, DistServe and SGLang contribute iteration-level scheduling, prefill/decode disaggregation and radix-tree prefix reuse [23, 24, 25]. Tutti and SARIA make SSD-backed KV practical with reload scheduling and persistence

selection [26, 27]. MeshStack borrows the premise and changes three things: tiers are devices in a home, movement is gated by authority as well as by cost, and mesh-wide prefill/decode disaggregation is rejected for generation 1 [2].

2.3 Lossless compression of KV and activations

Lossless KV compression is a young line: GAAC exploits bfloat16 exponent redundancy with calibrated per-channel codes for personal devices [28]; SplitZip targets disaggregated serving with ultra-fast codecs [29]; ZipFusion and others fuse decompression into attention. BloomBee uses a byte-split layout plus entropy coding for activations over the internet [13]. This paper does not design a codec. It measures the simplest construction in that family (byte planes plus a general entropy coder, Zstandard [30]) on real tensors so that the PLACE cost model has a measured ratio instead of a borrowed one, and it compares against a Gaussian control to show that the gain is a property of the tensors.

2.4 Schedulers, leases and fencing

Borg and Omega established capability-aware placement with shared state and optimistic concurrency [31, 32]; Chubby and Kubernetes leases established the renewable lease as the primitive that makes ownership disappear when its holder does [33, 34]. The Auto Device Role Mode note maps those lessons onto MeshStack: a metadata-only coordinator, a lease with a fencing epoch, hard filters before scoring, and a strict queue in which owner work is P0 [4]. The Dynamo KV Router and llm-d cost form that lineage B adopts prices prefill by prefix overlap and decode by load [2]. Kafila plans stage and memory placement on a trusted set of heterogeneous commodity machines [35]; its fleets are not churn-prone mobile devices.

2.5 Storage, offload and the edge

FlexGen and LLM in a flash established that disk and flash can extend capacity at a latency cost that depends on the demanded fraction of weights and on read layout [36, 37]; FlashMoE and the SSD-resident MoE work extend that to experts [38, 39]. CEMA-Router, Prism, Mesh-Attention and EasyBalance address routing, expert placement, attention locality and cross-layer balance at datacenter or edge-server scale [40, 41, 42, 43]; the lossless-transport note cites them as motivation [3], and this paper keeps them in that role: their numbers are theirs, on their hardware. The stage-level NPU study and the mobile battery study motivate recording dispatch, fallback and energy costs rather than selecting engines by advertised ratings [44, 45]. InferSlice's reference architecture for SLO-driven inference over a telco WAN is the closest formal treatment of SLO gates [46].

2.6 Positioning

System	Setting	Sharding form	KV movement	Authority model	Owner sovereignty, churn	Evidence class
prima.cpp [6]	home cluster, mixed CPU/GPU, Wi-Fi	pipelined ring with disk offload	none across nodes	none (trusted cluster)	not modelled	measured, authors' hardware
TPI-LLM [7]	low-resource edge devices	tensor	none	none	not modelled	measured, authors' testbed
EdgeShard / Galaxy [8, 9]	cooperating edge devices	layer partition / hybrid	none	none	not modelled	measured, authors' testbed
Petals, BloomBee [11, 13]	internet volunteers	pipeline	node-local	none (untrusted peers assumed)	checkpoint and replay	measured, authors' deployments
Mooncake, LMCache [18, 19]	datacenter	n/a	tiered, cost-driven	none	n/a	measured, production
Kafila [35]	trusted heterogeneous machines	pipeline, memory-aware	none	trusted set	not modelled	measured, authors' fleet
MeshStack lineage A [1]	THOX devices, SSD hosts	whole-request first; pipeline later	session-bound, ephemeral	leases per request	fail closed	calculated; protocol unexecuted
MeshStack lineage B [2]	THOX private mesh	contiguous pipeline (D10)	grant-gated T3 (D9)	secrets graph, consumption-time	replan or fail closed (D13)	reference implementation, plan level
Auto Device Role Mode [4]	THOX private mesh	whole-request routing	none	lease with fencing epoch	P0 owner work, drain	design note
This paper	THOX private mesh plus LoRa fallback	contiguous pipeline, one-stage degenerate case first	grant-gated, cost-gated, measured ratio	secrets graph with epoch-bearing lease grants	P0 owner work, fencing, consumption-time validity	measured on fleet host classes, calculated on floors, simulated protocol + live drill

Table 1. Design positioning. Evidence class names what each row's claims rest on; no row's measurements are quoted as THOX results.

3 System and threat model

3.1 Setting

A mesh is the set of nodes under one household's administrative control, reachable over a LAN or a private tailnet built on WireGuard [47]. Nodes are E2, P4 and P6 class devices with the public floors of Table 2 [5], plus at most a few larger hub nodes (a desktop, a ThoxKey host). Devices are personal: an owner may close, move or drain one at any moment, and a device that leaves is a routine event with a routine consequence. Links are shared and variable; a broadcast relay documented under roughly 50 peers is the current cross-site path [2]. A LoRa auxiliary mesh on ESP32-class radios running Meshtastic may be present; its over-the-air packet carries a 16-byte cleartext header and at most 239 bytes of encrypted payload, and its fastest preset is rated at 21.88 kbit/s [48, 49].

Class	Device line	RAM floor	Storage floor	On-device model class
E2	ThoxAir (wireless edge node)	2 GB	16 GB flash	about 1B at 4-bit
P4	ThoxMini (personal edge server)	4 GB	32 GB flash	about 3B, quantised
P6	ThoxClip (clip-on edge compute)	6 GB	64 GB flash	about 3B, quantised

Table 2. Public THOX compute-class floors as recorded in the ThoxOS whitepaper [5]. Floors are category-level definitions, not measurements of shipped hardware; no device is qualified.

3.2 Requirements

1. **R1 Capacity truth.** A plan admits only when the two-domain inequalities of Section 4.2 hold for every stage; there is no partial admission and Unplaceable is a value, not an exception.
2. **R2 Contiguity.** A stage is a contiguous layer range; the activation traffic per token is one frame per stage boundary.
3. **R3 Lossless wire by default.** Activation frames and KV bytes cross nodes bit-exact (raw f32 or bf16 with a lossless codec); a lossy transfer tier (FP8/INT8, decision D15 in lineage B) is a labelled opt-in that the benchmark wave must gate, never a default.
4. **R4 Chunk identity.** KV chunks are content-addressed by a parent-chained HMAC key bound to the principal's secret and the model fingerprint.
5. **R5 Authority at consumption.** A capability is valid only if a live path exists in the grant graph at the moment of use; there is no cached positive decision.
6. **R6 Fencing.** Every completion, activation frame and KV transfer carries the lease epoch of its session; anything older than the session's current epoch is rejected.
7. **R7 Owner sovereignty.** Hard filters (opt-in, mesh membership, model residency, memory reservation floor, owner foreground state, thermal band, battery policy)

run before any score; owner work is P0 and a mesh job never evicts it; an established lease drains rather than dies.

8. **R8 Fail closed.** Local-only requests never fall through to a hosted route; an Unplaceable plan or a dead grant path ends the session audibly.
9. **R9 Coordinator outside the payload path.** The coordinator stores routing metadata only; prompts, activations, KV bytes and weights never transit it.
10. **R10 Control-plane degradation.** Presence, lease heartbeats and revocation survive LAN loss on a carrier whose packets are at most 239 bytes; model and KV traffic does not use that carrier.

3.3 Threat model

In scope: an honest-but-curious peer in the same home that holds a channel key and sees headers and advertisements; a departed node that returns holding stale leases and stale decisions; a stolen or compromised device that retains ciphertext and keys for its own chunks; a LAN observer who sees WireGuard ciphertext and LoRa headers. Out of scope: a compromised coordinator host, a malicious engine binary on a certified node, side channels in timing of compression and routing, and physical attacks on key storage. Lineage A's warning stands: activations and KV expose input, and partitioning is not anonymisation [1, 11].

4 Design

4.1 Architecture

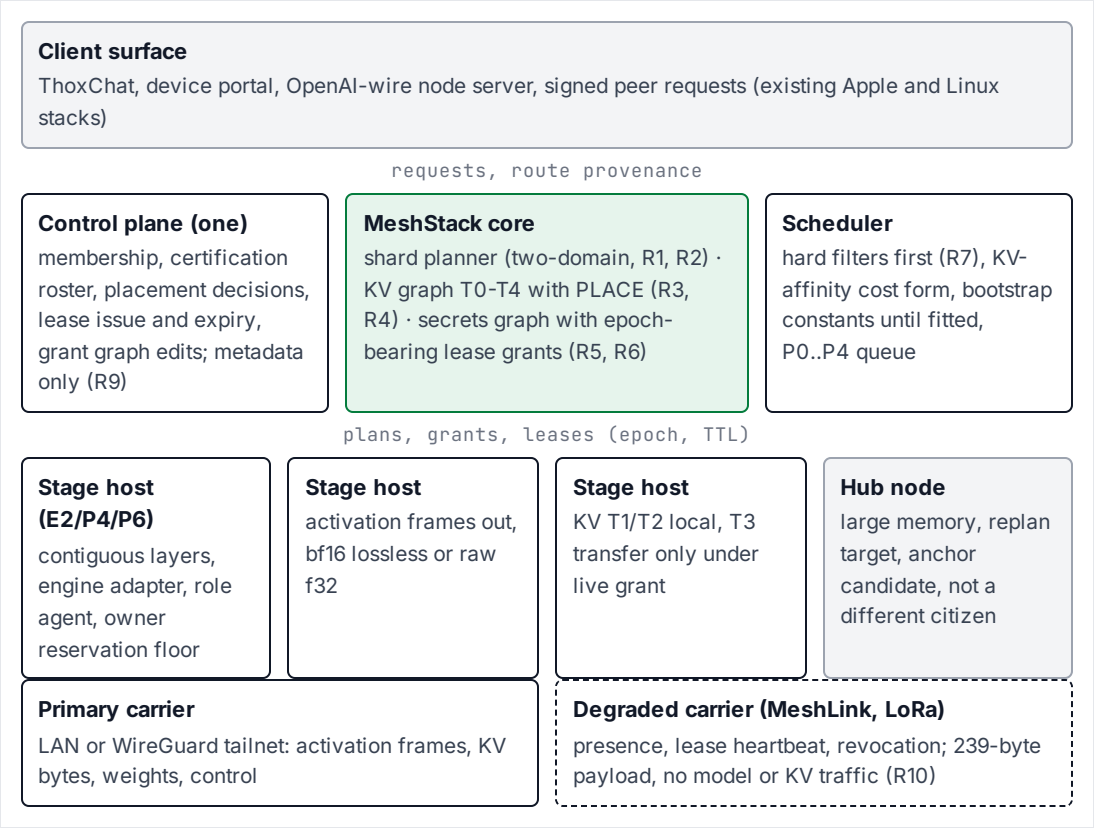


Figure 0. Reconciled architecture. The client surface and node servers are the existing stacks; one control plane issues decisions and leases; the MeshStack core holds the three mechanisms of lineage B; the scheduler holds the filters and queue of the role proposal; a LoRa carrier exists only for control-plane survival. Schematic, not a deployed topology.

4.2 Two-domain capacity admission

The sharder takes a model geometry (layer count, per-layer weight bytes at the target quantisation, per-layer KV bytes per token), a context policy T in tokens, a workspace estimate per stage, and a mesh state. For every stage s on node n the plan is admissible only if

$$W(s) + K(s)\,T \le R(n) \quad \text{and} \quad WS(s) \le \Sigma(n),$$

where $W(s)$ is the stage's weight bytes including the embedding on stage 0 and the head on the last stage, $K(s)$ is the stage's KV bytes per token summed over its attention layers, $R(n)$ is usable RAM, $WS(s)$ is priced host workspace and $\Sigma(n)$ is the node's system budget. The two domains are not summed. Lineage B records why this sentence is written so carefully: a merge that charged workspace against the RAM domain stranded 23 of 64 layers and failed four planner tests until

decision D23 restored the two-domain semantics [2]. Algorithm 1 is the first-fit contiguous planner used in this paper's calculations; it is conservative and reproducible, and the paper does not claim it is optimal. Whole-request placement is the case in which the first node admits every layer, so the existing routing path is a one-stage plan and ships first, as lineage A recommended [1].

Algorithm 1. First-fit contiguous two-domain planner

```
Input:  layers 0..L-1 with weight bytes w[l], kv bytes per token k[l]; context T;
        workspace WS; nodes n_0..n_m in order (anchor first) with R(n), Sigma(n)
Output: plan (list of stages) or UNPLACEABLE
l ← 0
for each node n in order:
    if WS > Sigma(n): continue                       # workspace domain
    start ← l; W ← embed if start = 0 else 0; K ← 0
    while l < L and W + w[l] + head(l) + (K + k[l]) * T ⩽ R(n):   # RAM domain
        W ← W + w[l] + head(l); K ← K + k[l]; l ← l + 1
    if l > start: emit stage(n, start, l-1, W, K*T, WS)
    if l = L: return plan
return UNPLACEABLE                                   # no partial admission (R1, R8)
```

4.3 Chunk identity: parent-chained HMAC keys

A KV chunk is the state for a span of tokens at one stage's layer range. Its key is

$$\kappa_0 = \mathrm{HMAC}_K\big(d \,\|\, 0^{32} \,\|\, f \,\|\, \tau_0 \,\|\, \ell(e)\big), \qquad \kappa_i = \mathrm{HMAC}_K\big(d \,\|\, \kappa_{i-1} \,\|\, f \,\|\, \tau_i \,\|\, \ell(e)\big),$$

with K the principal's keying secret, d a fixed domain string, f the 32-byte model fingerprint, τ_i the chunk's tokens as 4-byte big-endian integers and $\ell(e)$ a length-prefixed extra field. HMAC-SHA256 is the construction of RFC 2104 over FIPS 180-4 [50, 51]. Lineage B records that an earlier reference serialised tokens as decimal text and omitted the extra field, and that a two-language test against published vectors now pins the construction (decision D21) [2]; Appendix A gives this paper's own known-answer vector.

Proposition 1 (prefix sharing). Two sessions of the same principal and model whose first j chunks are identical produce identical keys $\kappa_0, \ldots, \kappa_{j-1}$. *Proof.* By induction on i: the inputs to the i-th HMAC are a deterministic function of $(K, f, \tau_0, \ldots, \tau_i)$, which the sessions share for $i < j$. □

Proposition 2 (no false hit from a divergent prefix). If HMAC-SHA256 with secret K is a pseudorandom function, then for sessions whose chunk sequences first differ at position j, the keys $\kappa_j, \kappa_{j+1}, \ldots$ collide with those of the other session with probability at most $q^2/2^{256}$ over q keys, and a different model fingerprint or a different principal gives independent keys at every position. *Proof sketch.* The PRF inputs differ in τ_j (or in f, or the secret differs), so the outputs are computationally independent of the honest session's keys; the chain propagates the difference

because κ_{i-1} is an input to κ_i. A collision would require a PRF output collision. □ Properties P1 to P5 in Section 6.2 are the executable form of these statements.

4.4 PLACE: local hit, transfer or recompute

When a session needs a chunk, PLACE answers from three options. A local hit costs nothing. A transfer of a prefix of L tokens with k bytes of KV per token over a link of effective payload rate B with per-transfer overhead a, using a lossless wire ratio r, and a recompute on a stage that prefills at P tokens per second, cost

$$T_{\text{xfer}}(L) = \frac{L\,k}{r\,B} + a, \qquad T_{\text{rec}}(L) = \frac{L}{P}, \qquad L^{*} = \frac{a}{1/P - k/(rB)} \ \text{ when } \tfrac{1}{P} > \tfrac{k}{rB},$$

and transfer wins for $L > L^{*}$; when the denominator is not positive, recompute wins at every prefix length. Transfer is additionally gated by authority (R5): a chunk moves only under a live grant path, however favourable the cost. Lineage B's cost form for session routing, $w_1 \,\text{prefill}\,(1 - \text{overlap}) + w_2 \,\text{decode load} + \ldots$ with thermal, battery, churn and link terms, is kept unchanged with its weights labelled bootstrap constants [2]. The ratio r is a measured per-model input (Section 6.1), not a constant.

4.5 Roles as epoch-bearing lease grants

The Auto Device Role Mode note proposes a renewable lease with fields that include a monotonic fencing epoch, an absolute expiry, a heartbeat interval, an owner-preemptible flag that must be false, and a fence token proving that inference belongs to the current epoch [4]. This paper folds that lease into lineage B's secrets graph rather than adding a second control plane: a lease is a grant edge from the coordinator principal to a node for a scope (a stage of a session, or a role class), carrying an epoch and a TTL. Validity is evaluated against the live graph at consumption (R5), so a departed node's leases die with its edges, and the epoch fences late arrivals (R6). The state machine is the note's: AVAILABLE, OFFERED, LEASED, DRAINING, back to AVAILABLE, with EXPIRED on timeout, disconnect or thermal emergency. Hard filters run before scoring, and the queue is P0 owner foreground, P1 owner local inference, P2 running leased completion, P3 new lease, P4 speculative work (R7).

Proposition 3 (no stale completion under fencing). If the coordinator increments the session epoch on every re-lease and a completion is accepted only when its epoch equals the session's current epoch, then no completion produced under a superseded lease is accepted. *Proof.* A superseded lease carries epoch e; the session's epoch is at least $e + 1$ from the re-lease onward and never decreases; equality cannot hold. □

Proposition 4 (bounded stale authorisation). With consumption-time validity, the number of accesses authorised after a grant-path node departs is zero; with a positive decision cache of lifetime C and consumption interval δ, it is at most $\lceil C/\delta \rceil$ per session. *Proof.* Live evaluation sees the missing edge at the first consumption after departure. A cached decision taken at time t_0 authorises consumptions until $t_0 + C$, at most $\lceil C/\delta \rceil$ of them. □ Section 6.6 measures both in simulation.

As deployed. The running coordinator implements the fencing of Proposition 3 without a separate epoch column: the fencing token is the unique, monotonically increasing lease identity issued from a persisted counter, a sweeper expires leases 5 s after their TTL lapses, and every mutating call (heartbeat, serving, complete, release) checks that the presented lease identity is the currently live one before it acts. A completion is therefore accepted only under the live lease identity, which is the acceptance property Proposition 3 states for the epoch form. The epoch-bearing lease record of the paragraph above is carried in the parallel app-facing schema (the v2 layer), not in the coordinator's own store. Section 6.6 reports a live drill of the deployed form.

4.6 Control plane and degradation to MeshLink

The control plane is one service (R9) that the record names sadie-mesh in lineage B and the Role Coordinator in the role note; the paper treats the name as a founder decision (Section 9) and specifies the behaviour. On the primary carrier it issues plans, grants and leases. When the LAN is gone, a MeshLink radio carries three messages: a presence beacon, a lease heartbeat and a revocation notice. Each fits in one Meshtastic packet. Meshtastic's channel traffic is AES-CTR under a shared key by default, with AES-CCM and signed broadcasts available on recent firmware and no forward secrecy [52]; the heartbeat therefore carries its own MAC under the lease key and the radio is never trusted for authority. Airtime is computed with the standard LoRa formula [53] in Section 6.5.

4.7 Security argument

Against an honest-but-curious peer: KV bytes move only under a grant the peer does not hold, advertisements are principal-keyed, positive-evidence only and bounded (lineage B's D19), and chunk keys reveal nothing about tokens without K. Against a departed node that returns: Propositions 3 and 4. Against a stolen device: crypto-shredding of the entry keys that made its chunks readable, so retained ciphertext is inert; this is a key operation, not a data hunt. Against a LAN observer: WireGuard on the primary carrier; on the LoRa carrier the observer sees the 16-byte header, which is why that carrier carries no content. Not argued here: a compromised coordinator, timing side channels in PLACE and the codec, and physical key extraction; Section 7 lists them.

5 Implementation

Two implementations exist. Lineage B's Python reference implementation (pipeline sharder, KV graph, secrets grant graph, KV-affinity scheduler, five frozen generation-1 wire schemas) has simulation semantics: it computes plans, keys, costs and grant decisions and executes no tokens; its root suite passed 63 tests with 44 subtests at HEAD b01bc03 on 2026-10-30 [2]. That repository is private and not attached to this work; its test states are quoted from the record and were not re-run here. Lineage A's calculator (33 scenarios, 10 unit tests) is attached and was re-run for this paper: 10 of 10 pass with 83 of 83 executable lines covered. This paper's own code is the experiment set of Appendix A: a clean-room planner, a clean-room keying implementation with property tests, the codec harness, the PLACE grid, the LoRa airtime calculator and the churn simulator, in stdlib Python plus NumPy, PyTorch (CPU) and the zstandard binding.

Requirement	Where specified	Test or evidence in this paper	Result
R1 capacity truth, R2 contiguity	Section 4.2, Algorithm 1	exp/admission_experiments.py: 54 scenarios, Unplaceable emitted with no partial plan	runs; results in Section 6.3
R3 lossless wire	Section 4.4	exp/compression_experiments.py: 28 bit-exact round trips (SHA-256 of decoded bytes)	all pass
R4 chunk identity	Section 4.3	exp/keying_experiments.py: properties P1-P5, known-answer vector	all pass
R5 authority at consumption	Section 4.5, Prop. 4	exp/churn_sim.py: stale authorisations, live vs cached	simulated + live drill; Section 6.6
R6 fencing	Section 4.5, Prop. 3	exp/churn_sim.py: stale completions, fenced vs unfenced	simulated + live drill; Section 6.6
R7 owner sovereignty	Section 4.5	not tested here; filters and queue are specification	unbuilt
R8 fail closed	Algorithm 1 last line	admission scenarios that return Unplaceable	runs
R9 coordinator outside payload	Section 4.1	architectural; lineage B schemas carry no KV bytes in T4 manifests	specification
R10 control-plane degradation	Section 4.6	exp/lora_experiments.py: airtime and heartbeat capacity	calculated; Section 6.5

Table 3. Requirement-to-test map. 'Runs' means the script executes and its outputs are in summary.json; 'simulated' and 'calculated' are evidence classes, not measurements on devices.

6 Evaluation

Setup. The v1.0.1 harness (Sections 6.1 baseline, 6.2, 6.3, 6.5, 6.6 simulation) ran on the authoring container, Linux, 2 vCPU, 7 GB RAM, no accelerator, Python 3.13.18, PyTorch 2.15.0+cpu (CPU), with Qwen/Qwen2.5-0.5B-Instruct [54, 55]: 24 layers, 2 KV heads of dimension 64, so 12,288 bytes of bf16 KV per token (48 MiB at 4096 tokens). The v1.1.0 Virtual Lab measurements ran on fleet host classes, labelled at each result: the codec on an Azure VM (CPU-only, 4 vCPU), the transfer between a fleet GPU host (RTX 4060 Ti) and an Azure VM, the engine split between a CPU VM client and an EC2 Mac (Apple M4, Metal), and the fencing drill live on the deployed coordinator (Azure VM host class). Timing in the harness used one warm-up then 30 repetitions per configuration, medians with interquartile ranges. Every number below is read from summary.json; every figure is generated from it; the lab sections record the sha256 of the lane result file each number traces to.

6.1 How much does a lossless codec buy on the wire? (1.17x to 1.19x on real GGUF KV)

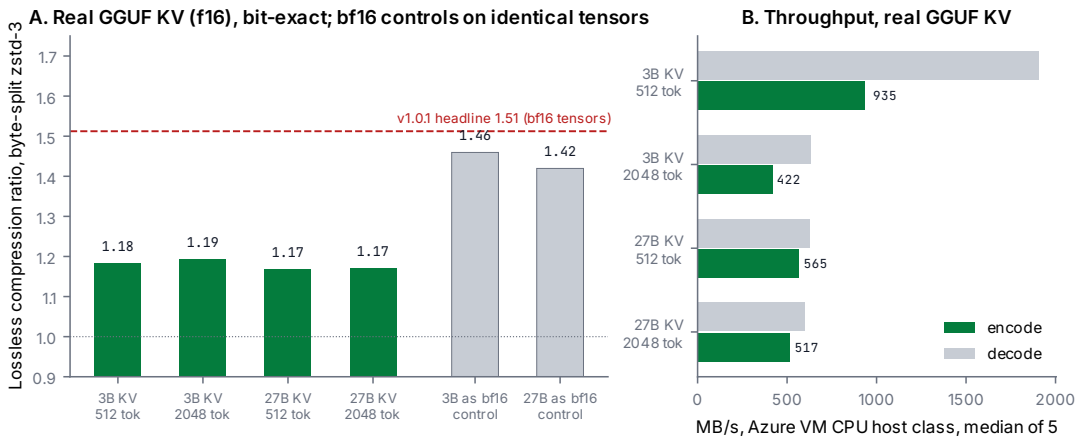


Figure 1. Lossless compression of KV state captured from real GGUF inference runs (llama.cpp state capture, f16), byte-split zstd-3, Azure VM CPU host class. A: ratio per sample; every bar is a verified bit-exact round trip. The dashed line is the v1.0.1 headline (1.51x), measured on bfloat16 tensors in the harness, not on captured engine state. B: encode and decode throughput for the same samples.

Sample (real GGUF KV, f16)	Tokens	Ratio	Encode MB/s	Decode MB/s	Bit-exact
3B attention KV	512	1.183	935	1,908	yes
3B attention KV	2,048	1.195	422	635	yes
27B attention KV	512	1.169	565	627	yes
27B attention KV	2,048	1.171	517	599	yes
27B recurrent (SSM) state, supplementary	fixed per sequence	1.107	490	552	yes

Table 4. Byte-split zstd-3 on KV bytes captured from real GGUF inference (Qwen2.5-3B and the 27B hybrid model), Azure VM (CPU-only, 4 vCPU) host class. Five timing repetitions per sample. Source: exp/results/vlab_codec.json.

On KV captured from a running engine, the byte-split zstd-3 codec reaches 1.17x to 1.19x, bit-exact on every sample (the byte-split maximum across levels is 1.20x, at level 9 on the 3B sample; summary statements in this paper use the cross-level range), at 422 to 935 MB/s encode. That is the number PLACE and the T3 transfer tier should use for GGUF-class state on a CPU host class. It is materially below the 1.51x headline of v1.0.1, and the difference is now explained rather than asserted: v1.0.1 measured tensors serialised in bfloat16 from a Hugging Face model in the harness. Casting the identical captured tensors to bfloat16 and re-running the same codec gives 1.42x (27B) to 1.46x (3B): tensor precision is the dominant term in the gap, because f16 engine state spreads its exponents more widely than bf16 serialisation does. The codec did not change between the editions.

The v1.0.1 harness results stand as harness results. On its 2048-token bfloat16 KV sample, zstd-3 on the raw byte stream reaches 1.31x; the byte-split form lifts that to 1.51x at level 3 and 1.69x with bytesplit-lzma, which is too slow to use; the high byte plane alone compresses 5.81x. Layer-boundary activations compress 1.43x at 2048 tokens in the same harness. A unit Gaussian bfloat16 stream of the same size compresses 1.41x, within 7 percent of the harness KV result: values whose magnitudes cluster within a few binades use few exponent codes, and both real bfloat16 activations and a unit Gaussian do. Three consequences follow. A lossless tier halves nothing, so PLACE must use a measured ratio per model and per precision, and the lossless-transport note's expectation of large lossless gains is not supported [3]. A model-specific gain beyond the Gaussian baseline exists but is small, so calibrated codecs of the GAAC kind are where further lossless gain would come from [28]. And at 422 MB/s and above the codec is not the bottleneck on any LAN a household owns, so compression is on by default for T3 transfers and off only on links faster than the encoder.

6.2 What does chunk keying cost? (9.7 microseconds per 64-token chunk)

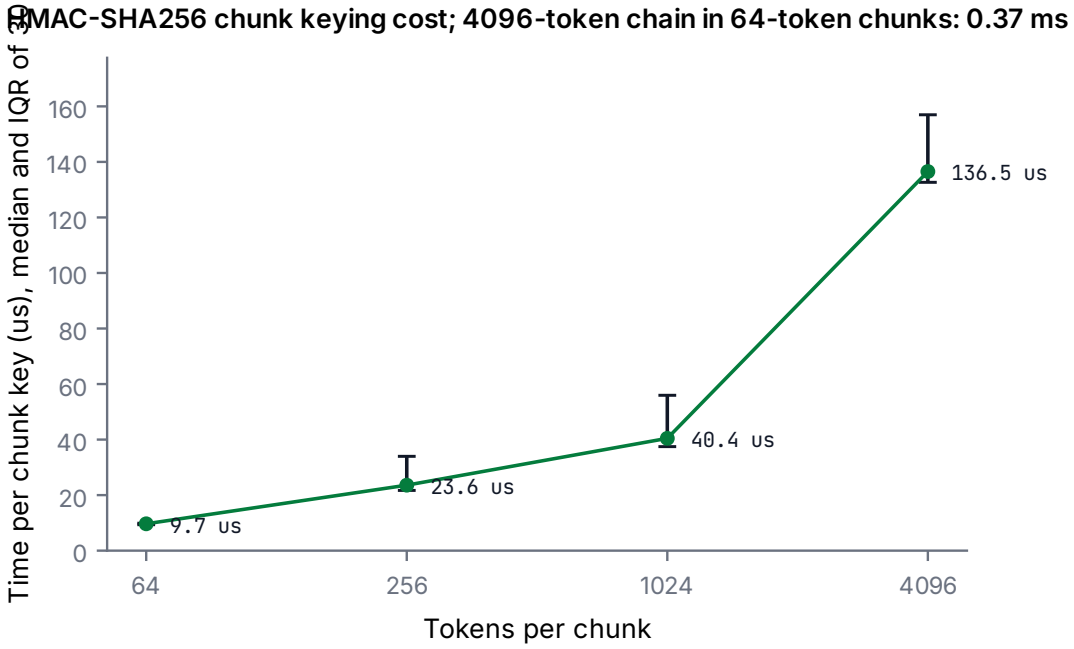


Figure 2. Cost of one parent-chained HMAC-SHA256 chunk key by chunk size (median and IQR of 30), stdlib implementation on the authoring host. The 4096-token chain cost is the title's second number.

Keying a 64-token chunk costs 9.7 microseconds (103,434 keys per second on one core); a 4096-token prefix keyed in 64-token chunks costs 0.37 ms end to end. Against a prefill that takes seconds and a transfer that takes tens of milliseconds, keying is noise, and there is no reason to batch or cache it. Properties P1 to P5 (prefix sharing, divergence, model binding, principal binding, determinism) all pass; the script fails if any does not. The known-answer vector in Appendix A lets the demo page and any other implementation check the same construction.

6.3 Does a trio at the public floors place the 27.8B demo model? (No)

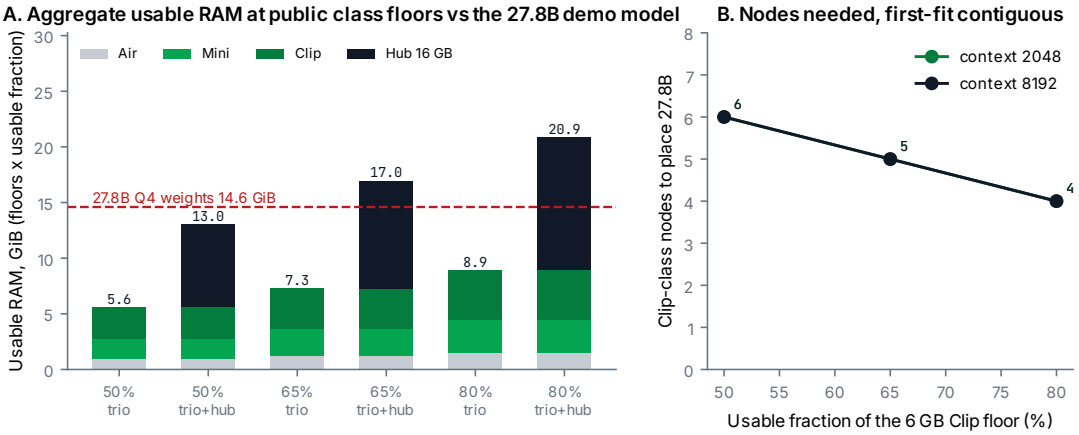


Figure 3. A: aggregate usable RAM of an Air+Mini+Clip trio at the public floors (2, 4, 6 GB) for three usable fractions, with and without a 16 GB hub, against the 14.6 GiB of Q4 weights in the reference demo geometry. B: Clip-class nodes needed to place that geometry with first-fit contiguous placement. Calculated from floors and recorded geometry; nothing measured.

The reference implementation's demo places the 27.8B model on three nodes because its fixtures give the anchor 9.6 GiB usable [2]. At the public floors the trio holds 12 GB in total, less than the 14.6 GiB of weights before any KV reserve or usable fraction, so the plan is Unplaceable at every fraction and every context we tried; adding a 16 GB hub places it at 50% (Unplaceable), 65% (placed), 80% (placed). Without a hub the geometry needs 6 Clip-class nodes at 50% usable and 2048 tokens of context, and 4 at 80% and 8192 tokens. The smaller models behave as the ThoxOS whitepaper's capacity section predicts [5]: the published 2.49 GB ThoxMini-Phi4Mini file is Unplaceable on one Air at 2048 tokens and placed on one Mini; at 32768 tokens it is Unplaceable on one Mini and Unplaceable on Mini+Air, which is the regime where a two-stage plan buys context rather than parameters. The 0.5B model is Unplaceable on one Air at 32768 tokens and 65% usable. This is the paper's central negative result and it changes the product statement: generation-1 MeshStack serves the 27B class only with a hub in the mesh, and the benchmark wave's bench trio must be described by its real memory, not by the fixtures.

Corrected state geometry (v1.1.0 recompute). The v1.0.1 admission analysis carried a state figure of 188,367 B/token for the 27B class, taken from the design record. Lane T2 measured the real GGUF state directly: attention KV is 65,536 B/token, and the hybrid model's linear-attention layers hold a fixed recurrent state of 156,894,356 B per sequence that does not grow with the prefix (total measured sequence state: 190,461,492 B at 512 tokens and 291,161,652 B at 2048 tokens). The old figure overstated the per-token term by about 2.9x and omitted the fixed state entirely. We recomputed the affected admission outputs with the paper's own planner (exp/admission_experiments.py, corrected model in summary.json): the trio

outcome is Unplaceable at every usable fraction, the trio-plus-hub outcome is Unplaceable at 50%, placed at 65%, placed at 80%, and the Clip counts are 6 at 50%, 5 at 65%, 4 at 80% (2048-token context), identical to the v1.0.1 outcomes, because the admission boundary is set by the 14.6 GiB of weights against 12 GB of floors, not by the state term. Any admission output computed anywhere with the 188,367 B/token figure is superseded by this recompute.

6.4 When does transfer beat recompute? (Measured)

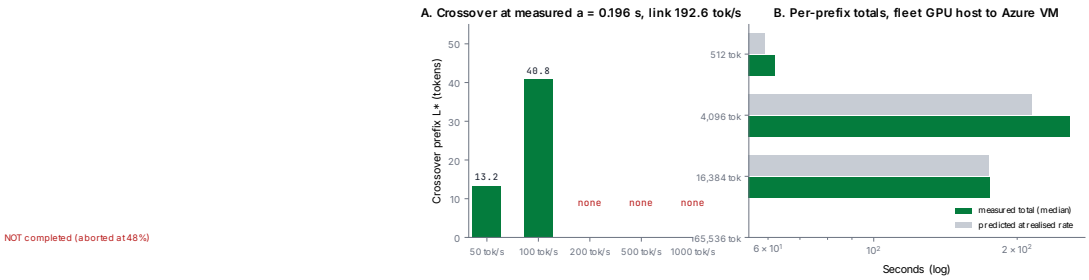


Figure 4. PLACE with measured inputs. A: crossover prefix for the 27.8B geometry at the measured link token rate (192.6 tok/s) and the measured per-transfer overhead a = 0.196 s, by assumed prefill rate P. A missing bar means recompute wins at every prefix length. B: per-prefix transfer totals, measured against the PLACE prediction at each run's realised link rate. The 65536-token prefix was not completed. Transfer measured between a fleet GPU host and an Azure VM; bundles are the synthetic bfloat16 Gaussian construction at the 27.8B geometry, not real model KV.

Prefix (tokens)	Measured total (median)	Predicted at realised rate	Error	Status
512	61.9 s	58.9 s	-4.8%	completed, hash-verified
4,096	257.8 s	213.8 s	-17.0%	completed, hash-verified
16,384	175.4 s	174.3 s	-0.6%	completed, hash-verified
65,536	not claimed	not claimed		NOT completed: aborted at 48% received after about 80 minutes on a stalled path

Table 5. Grant-gated KV transfer between a fleet GPU host (Windows/WSL2, RTX 4060 Ti) and an Azure VM (Linux) over a private overlay link. Source: exp/results/vlab_transfer.json.

v1.0.1 evaluated PLACE on assumed inputs, including a per-transfer overhead a = 0.02 s that nobody had measured. Lane T3 measured the real path: the overhead is a = 0.196 s (median of 7 one-KiB grant sessions; connect, grant wait and completion return, about three round trips at the measured round-trip time), ten times the assumption. The bulk link was equally unlike a constant: median receiver payload 71.5 Mb/s with repetitions at 79.1, 15.5 and 71.5 Mb/s on the same path, and long transfers sagged far below the short-window rate. The transferred bundles were this paper's synthetic bfloat16 Gaussian construction at the 27.8B geometry (measured bundle ratio 1.41), not real model KV; the protocol, grant path and byte counts are the measured objects.

With the measured overhead, the measured link token rate (192.6 tok/s at the measured ratio) and the 27.8B geometry, the crossover is 13.2 tokens at a 50 tok/s prefill and 40.8 tokens at 100 tok/s, and there is no crossover at 200 tok/s or above: a stage that prefills faster than the link's token rate should always recompute, which is the rate comparison of Section 4.4 with measured constants. Judged per run, PLACE with each run's realised rate predicts the measured totals to within 0.6 to 17 percent (Table 5); judged against a single assumed link rate it fails by an order of magnitude on the sagged runs, which is the operational lesson: PLACE needs the telemetry of lineage B's D13, not a better constant [2]. The 65536-token transfer is recorded as not completed and no number is claimed for it.

6.5 What can a LoRa control plane carry?

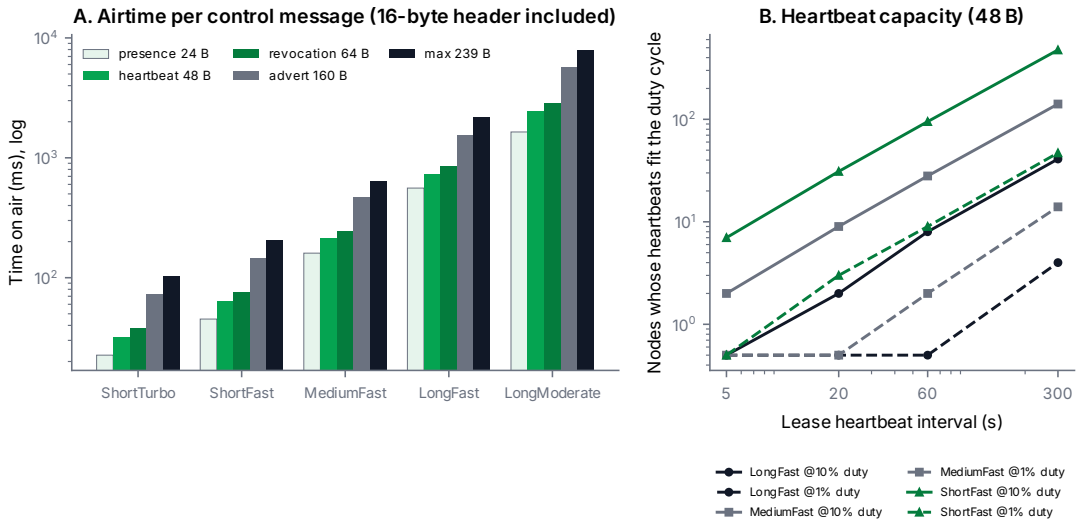


Figure 5. A: time on air for each MeshStack control message on five Meshtastic presets, 16-byte header included, standard LoRa airtime formula with a 16-symbol preamble. B: number of nodes whose 48-byte lease heartbeats fit a 10% or 1% duty cycle at each heartbeat interval. Calculated; no radio measured.

A 48-byte lease heartbeat takes 723 ms on LongFast and 63 ms on ShortFast; a full 239-byte packet takes 2,157 ms and 204 ms. Under a 10% duty cycle LongFast carries 20-second heartbeats for 2 nodes and ShortFast for 31; under 1% LongFast carries them for 0 nodes at 20 seconds and 4 at 300 seconds, and ShortFast 0 at 5 seconds. A household mesh of a dozen nodes fits on the slow presets only with heartbeats of a minute or more, which is adequate for presence and revocation and inadequate for the 5-second heartbeat the role note proposes for the primary carrier [4]; the degraded mode therefore stretches lease TTLs rather than fail. For scale, moving the 0.5B model's 4096-token KV (48 MiB) over ShortFast at a 10% duty cycle would take about 12 hours: the carrier is a control plane and nothing else (R10). Duty cycles are region specific and the Meshtastic firmware is GPL-3.0, which ThoxOS keeps out of its Core linking boundary [5]; MeshLink is therefore a separate device, not a ThoxOS component.

6.6 Do fencing and consumption-time validity hold under churn? (Simulation + live drill)

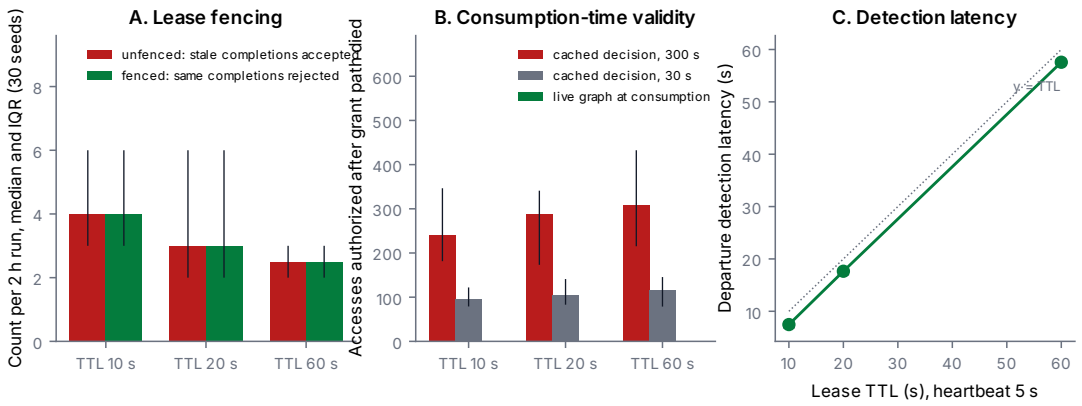


Figure 6. Discrete-event simulation, 6 nodes, 2-hour runs, 30 seeds, mean uptime 600 s, heartbeat 5 s. A: stale completions an unfenced protocol would accept versus the same completions rejected by epoch fencing. B: accesses authorised after the grant-path node departed, for two decision-cache lifetimes and for live evaluation. C: departure detection latency against lease TTL. Medians with IQR. Protocol logic only; the live drill below is the deployed-system counterpart.

With a 20-second TTL and 30-second decision cache the median run has 356 sessions and 16 re-leases; an unfenced protocol accepts 3 stale completions from returning holders and the fenced protocol rejects all 3 of them, as Proposition 3 requires. Live evaluation authorises 0 accesses after a grant-path node has left and fails 18 sessions closed at their next consumption; a 30-second cache authorises 105 accesses per run with at most 14 in one session (the Proposition 4 bound at a 2-second consumption interval is 15), and a 300-second cache 288 with at most 50, where the session's own remaining service time, not the cache, is the binding limit. Detection latency tracks the TTL: 7.5 s at 10 s, 17.6 s at 20 s, 57.6 s at 60 s, which is the trade the role note names: a shorter TTL finds departures faster and costs more heartbeats, and the LoRa budget of Section 6.5 bounds how short it can be in degraded mode. These are properties of the protocol under stated distributions; they say nothing about how often real devices leave.

Live drill (measured). The same acceptance property was then exercised on the deployed production coordinator (Azure VM host class), using two drill devices and no production lease. Holder A was granted lease-000015 and marked serving; A then departed mid-session. The lease expired at its 300 s TTL and the coordinator logged the expiry about 0.5 s after the lapse, consistent with its 5 s sweeper and with the simulation's claim that detection latency tracks the TTL. Holder B was granted lease-000016. Every stale action attempted under lease-000015 (heartbeat, serving mark, completion) was rejected with HTTP 404; stale authorisations accepted: 0; stale completions accepted: 0. The live holder's completion under lease-000016 was accepted, and the pool drained to zero leases. The drill validates the deployed mechanism of Section 4.5 (a monotonic

lease identity checked on every call), which is the form the fleet actually runs; the epoch-bearing form remains the specification for the app-facing v2 schema. Source: exp/results/vlab_roles.json.

6.7 Can a stock engine split a model across two hosts? (Measured, n = 1)

Configuration	Host classes	Generation	Prompt processing	Runs
Two-host split, layer 2,1	CPU VM client + EC2 Mac (Apple M4, Metal) server	2.3 tok/s	18.1 tok/s	1 completed of 17 attempts
Single host (the split client)	CPU VM, heavily contended	0.9 tok/s median		3
Single host (reference)	Azure VM, CPU-only, uncontended	21.0 tok/s median		3

Table 6. llama.cpp (ggml-rpc) split of Qwen2.5-3B Q4_K_M, context 2048, 128 generated tokens, seed 42. The two single-host rows are different host classes and are not interchangeable baselines. Source: exp/results/vlab_engine.json.

A stock llama.cpp build split Qwen2.5-3B Q4_K_M across two hosts (layer split 2,1, ggml-rpc tunnelled over SSH because both clouds' firewalls block the RPC ports) and generated at 2.3 tokens per second, against 0.9 tok/s for the same model on the client host alone: offloading two thirds of the layers to the remote Metal GPU beat the contended local CPU despite the wide-area path. The run took 375 s of wall time including tensor upload. The caveats are part of the result: exactly one split run completed out of 17 attempts across topologies, and every failure was transport (tunnel session lifetime, receive-window collapse, server session death), not engine; the number is an existence proof with a throughput indication, not a benchmark. The 21.0 tok/s single-host figure on an uncontended Azure VM is a different host class and is reported only so the split is never compared against it.

Generation-phase wire bytes per token were NOT isolated: on the completed run the client socket had closed before the counter snapshot ran, so load-phase and generation-phase bytes could not be separated, and this paper claims no measured wire rate for generation. The calculated boundary term is 4,096 B per generated token (hidden size 2048 in fp16 crossing the split boundary once per token, plus small control messages), labelled CALCULATED. The 27B split was not achieved: the GPU host class was unreachable for the whole session, and the 27B model as a single-host client on the 7.9 GB VM was killed by the out-of-memory killer with zero tokens generated. Both facts are recorded, not worked around.

7 Discussion and limitations

The section a sceptical reader should read first.

1. **Hosts and models.** No number in this paper was measured on a THOX device. The v1.0.1 harness ran on one 2-vCPU container with one 0.5B model in fp32 with bfloat16 serialisation; the Virtual Lab measurements ran on named general host classes (Azure VM CPU, fleet GPU host, EC2 Mac Apple M4, the deployed coordinator). Ratios and throughputs for other models, quantisations, precisions or hosts must be measured; the codec ratio in particular is a per-model, per-precision input.
2. **Repetitions and statistics.** Timing used one warm-up and 30 repetitions per configuration, reported as medians with interquartile ranges; no confidence intervals or significance tests are computed. The simulation used 30 seeds of 2 simulated hours.
3. **Engine and mesh evidence is thin.** One two-host split run completed (n = 1) over tunnelled RPC, with 16 of 17 attempts killed by transport failures; generation-phase wire bytes per token were not isolated; the 27B split was not achieved (GPU host unreachable, single-host attempt out of memory). The cross-host transfer path showed minute-scale rate variance, and the 65536-token transfer was not completed. No stage executed on a THOX device, and no throughput, latency or energy figure for MeshStack on devices exists.
4. **Calculated inputs.** The admission results rest on public floors, a usable-fraction sweep, a 256 MiB workspace assumption and uniform per-layer weight splits for the 27.8B and Phi-4-mini geometries; the 27B state geometry is now measured (Section 6.3) and the PLACE overhead and link terms are measured on one fleet path (Section 6.4), but prefill rates for THOX devices still do not exist, so the crossover table is evaluated at assumed prefill rates. The LoRa results rest on the standard formula, the documented 16-symbol preamble and illustrative duty cycles.
5. **Not built.** Owner-sovereignty filters and the P0..P4 queue, the role agent, production endpoint authentication, real key custody, crypto-shredding and erasure, and the MeshLink message formats are specification. Lineage B's reference implementation was not re-run; its test states are quoted.
6. **Codec scope.** The byte-split zstd codec is a baseline from the literature, not a contribution; GAAC-style calibrated codes and fused decompression were not evaluated.
7. **Security scope.** No adversarial evaluation of the grant graph, inventory keying or frame scoping has been run; timing side channels in PLACE and the codec are unexamined.
8. **Control-plane ownership.** Three documents name three control-plane owners; this paper specifies behaviour and leaves the name to Section 9.
9. **Record motion.** The ThoxOS whitepaper (2026-10-31) states that no kernel-model weights exist, while the kernel-model cards dated 2026-11-01 report gate

records; that is a one-day staleness in a cited source, not a finding of this paper, and it does not affect any result here.

10. **Not peer reviewed.** This is a preprint prepared by THOX.ai Research with AI assistance (Appendix C). External submission needs explicit founder approval.

8 Conclusion

MeshStack's two design lineages agree more than their documents suggest: pipeline sharding with fail-closed two-domain admission, a KV graph whose transfers are gated by cost and authority, and routine churn handled by replanning. This paper writes the agreement down once, adds the lease and role layer as epoch-bearing grants inside the same secrets graph, and names the LoRa carrier's job. The measurements say the lossless wire is worth 1.17x to 1.20x on real GGUF KV and the keying is free; the transfer, fencing and engine measurements of Sections 6.4, 6.6 and 6.7 replace the last assumed constants with measured ones, caveats included. The calculation says the current demo geometry does not fit the devices the company sells without a hub, which the product statement must now reflect. The simulation says the fencing and consumption-time rules do what their propositions claim. What remains is the evidence that needs a fleet, and the paper states the gate it must pass.

9 Roadmap and decisions for the founders

Ordered by what unblocks what. Items 1 to 3 are decisions; the rest is work.

1. **Control-plane name and owner.** Lineage B names sadie-mesh canonical (DECIDE-01); the role note embeds a Role Coordinator beside routing; THOX_MeshStack ships a Rust coordinator. One service, one name; the lease-grant unification here lets any of the three carry it.
2. **Product statement for generation 1.** Device-plus-hub for the 27B class (Section 6.3); device-only for the 3B class with context as the lever. The benchmark wave's bench trio must be described by its real memory.
3. **Authorship and venue.** arXiv does not permit AI systems as authors. The roster for this edition is settled: Craig Ross and Tommy Xaypanya are the authors, with Tommy Xaypanya as corresponding author; no AI system is an author.
4. **Measure the ratio per model, on device.** Done for real GGUF KV on a CPU host class in this edition (Section 6.1); the remaining step is the same measurement on a Mini-class device, and a measured ratio recorded per model wherever one enters the catalog.
5. **Bench trio with real floors.** First real pipeline run of a 3B class model across Air, Mini and Clip at their floors with activation telemetry (per-hop latency, per-stage compute) as the exit gate; then the 27B class with a hub.

6. **Fit the scheduler weights.** Replace the bootstrap constants of the cost form with values fitted from the telemetry; print the losing regime (short, cold, unique requests with a capable hub present).
7. **MeshLink message spec.** Freeze presence, heartbeat and revocation formats under 239 bytes with a lease-key MAC; test on two ESP32 radios; record the regional duty cycle.
8. **Adversarial pass.** Grant graph, inventory keying, frame scoping and timing channels, before any public performance claim.

A Appendix A. Reproducibility

Environment: Linux container, 2 vCPU, 7 GB RAM; Python 3.13.18; PyTorch 2.15.0+cpu (CPU wheel); transformers 5.22.1; zstandard; NumPy; Matplotlib. Node 22 with katex 0.16.11, @fontsource/inter 5.1.0, @fontsource/jetbrains-mono 5.1.1 and Playwright for rendering.

Commands (from the artifact root)

```
pip install torch --index-url https://download.pytorch.org/whl/cpu && pip install
transformers safetensors zstandard numpy matplotlib pypdf
python3 exp/compression_experiments.py   # downloads Qwen/Qwen2.5-0.5B-Instruct; ~15 min
on 2 vCPU
python3 exp/keying_experiments.py        # fails on any property violation
python3 exp/admission_experiments.py
python3 exp/place_experiments.py         # reads the measured ratio from
compression.json
python3 exp/lora_experiments.py
python3 exp/churn_sim.py                 # 30 seeds x 18 configurations, ~2 min
python3 exp/vlab_evidence.py             # verifies lane hashes; writes the scrubbed
vlab_*.json (needs the lab tree)
python3 exp/analyze.py                   # writes exp/results/summary.json
python3 wp/figures.py && python3 wp/build.py && node wp/katex_render.cjs
wp/paper.src.html wp/paper.html
node wp/pdf.cjs wp/paper.html wp/cover.pdf wp/body.pdf && python3 wp/merge_pdf.py
```

Virtual Lab evidence: the lane result files live in the THOX Virtual Lab tree and are not attached; exp/vlab_evidence.py verifies their sha256 (two are hash-pinned in the script) and writes the scrubbed public-edition copies exp/results/vlab_codec.json, vlab_transfer.json, vlab_roles.json and vlab_engine.json, each recording the sha256 of the lane original it derives from and using host classes only. analyze.py folds them into summary.json, so every v1.1.0 number traces to the same single evidence source.

Known-answer vector for the keying construction (secret = bytes 0x00..0x1f, fingerprint = 32 bytes of 0xab, chunks [1,2,3,4] then [5,6]):

```
key_0 = 57583d1c59f44e361985646b98a63933e762bdb19f1734f866873953cb85ecb2
key_1 = 51be4cd6edb0062c897751ab9e65f3a53626788d8f4d0b5edb3e8c532caefa14
```

Prompt SHA-256 for the 2048-token sample: `b4c3d8a9912eef7b4acb502d6dde047396aac441d74734dc2762610f1d46b6b9` (8.0 repeats of the base passage; host prefill 14.1 s). The artifact zip contains every script, raw results JSON, summary.json, figure sources and the built outputs.

B Appendix B. Notation

Symbol	Meaning
$W(s), K(s), WS(s)$	stage weight bytes, KV bytes per token, priced workspace bytes
$R(n), \Sigma(n)$	node usable RAM and system (workspace) budget
T	context policy in tokens
κ_i, K, f, τ_i	chunk key, principal keying secret, model fingerprint, chunk tokens
L, k, r, B, a, P	prefix tokens, KV bytes per token, lossless ratio, link payload rate, transfer overhead, prefill rate
L^*	PLACE crossover prefix
C, δ	decision-cache lifetime, consumption interval
E2, P4, P6	THOX compute classes (Air, Mini, Clip)
T0-T4	KV tiers: in-context, node-local memory, node-local persistent, mesh transfer, manifest

C Appendix C. Authorship and disclosure

This preprint is issued by THOX.ai Research. The authors are Craig Ross (Chief Executive Officer) and Tommy Xaypanya (Chief Technology Officer and Co-Founder), THOX.ai LLC; Tommy Xaypanya is the corresponding and accountable human author. Author consent to this edition is recorded before any submission. Generative AI assistance: the literature retrieval, experiment scripts, analysis, figures and draft text were produced by an AI system (Claude, Anthropic) operating as the THOX digital human Odette Lark under the direction of the corresponding author; every citation was resolved live against its primary page on 2026-11-03 and every number is generated from the experiment outputs. No AI system is an author. Conflicts: the authors are THOX.ai principals; this is an interested-party technical account. Edition history: v1.0.0 was an internal confidential draft; v1.0.1 (2026-11-03) was the public preprint edition, deposited at Zenodo (version DOI 10.5281/zenodo.23919580). This edition (v1.1.0) is the evidence edition: it corrects the codec ratio (Section 6.1), the 27B state geometry and the affected admission outputs (Section 6.3) and the PLACE overhead and crossover analysis (Section 6.4) against Virtual Lab measurements; corrects the fencing mechanism description to the deployed form and adds its live drill (Sections 4.5, 6.6); and adds the first two-host engine measurement (Section 6.7).

The design, the negative capacity result and the protocol propositions are unchanged from v1.0.1. Cite this work by the Zenodo concept DOI 10.5281/zenodo.23919579; the version DOI for v1.1.0 is pending deposit.

References

[1] Craig Ross, Tommy Xaypanya. *Storage Aware Parallel Inference for THOX MeshStack Devices*. THOX.ai LLC internal document THOX-RP-MESH-2026-1101, edition 1.1.0, 2026-11-01 (confidential, unpublished).

[2] THOX.ai. *MeshStack v2: Distributed Inference for the Private Device Mesh (design paper v1.0.0)*. THOX.ai internal design paper, 2026-10-30 (unpublished; records decisions D1-D23, the Qwen3.6-27B demo geometry and the fleet engine spike).

[3] THOX.ai. *MeshStack v2: Architecture and Methodology for Lossless Distributed Inference in Private Local AI Meshes (draft note)*. THOX.ai internal draft, October 2026 (unpublished).

[4] THOX.ai. *Auto Device Role Mode for THOX MeshStack (research note)*. THOX.ai internal research note, October 2026 (unpublished).

[5] THOX.ai. *ThoxOS: An Open Kernel and Operating System for Local-First AI (technical whitepaper v0.1.0)*. THOX.ai, 2026-10-31 (unpublished; source of the E2/P4/P6 class floors and the open/paid line).

[6] Zonghang Li, Tao Li, Wenjiao Feng, Rongxing Xiao, Jianshu She, Hong Huang, Mohsen Guizani, Hongfang Yu, Qirong Ho, Wei Xiang, Xue Liu. *Prima.cpp: Fast 30-70B LLM Inference on Heterogeneous and Low-Resource Home Clusters*. arXiv:2504.08791v3, first posted 2025-04-07, version inspected 2026-07-04. https://arxiv.org/abs/2504.08791

[7] Zonghang Li, Wenjiao Feng, Mohsen Guizani, Hongfang Yu. *TPI-LLM: Serving 70B-scale LLMs Efficiently on Low-resource Edge Devices*. arXiv:2410.00531v1, first posted 2024-10-01, version inspected 2024-10-01. https://arxiv.org/abs/2410.00531

[8] Mingjin Zhang, Jiannong Cao, Xiaoming Shen, Zeyang Cui. *EdgeShard: Efficient LLM Inference via Collaborative Edge Computing*. arXiv:2405.14371v1, first posted 2024-05-23, version inspected 2024-05-23. https://arxiv.org/abs/2405.14371

[9] Shengyuan Ye, Jiangsu Du, Liekang Zeng, Wenzhong Ou, Xiaowen Chu, Yutong Lu, Xu Chen. *Galaxy: A Resource-Efficient Collaborative Edge AI System for In-situ Transformer Inference*. arXiv:2405.17245v1, first posted 2024-05-27, version inspected 2024-05-27. https://arxiv.org/abs/2405.17245

[10] Junchen Zhao, Yurun Song, Simeng Liu, Ian G. Harris, Sangeetha Abdu Jyothi. *LinguaLinked: A Distributed Large Language Model Inference System for Mobile Devices*. arXiv:2312.00388v1, first posted 2023-12-01, version inspected 2023-12-01. https://arxiv.org/abs/2312.00388

[11] Alexander Borzunov, Dmitry Baranchuk, Tim Dettmers, Max Ryabinin, Younes Belkada, Artem Chumachenko, Pavel Samygin, Colin Raffel. *Petals: Collaborative Inference and Fine-tuning of Large Models*. arXiv:2209.01188v2, first posted 2022-09-02, version inspected 2023-03-02. https://arxiv.org/abs/2209.01188

[12] Alexander Borzunov, Max Ryabinin, Artem Chumachenko, Dmitry Baranchuk, Tim Dettmers, Younes Belkada, Pavel Samygin, Colin Raffel. *Distributed Inference and Fine-tuning of Large Language Models Over The Internet*. arXiv:2312.08361v1, first posted 2023-12-13, version inspected 2023-12-13. https://arxiv.org/abs/2312.08361

[13] Jiu Chen, Shuangyan Yang, Xu Xiong, Hexiao Duan, Xinran Zhang, Jie Ren, Dong Li. *Distributed Generative Inference of LLM at Internet Scales with Multi-Dimensional Communication Optimization*. arXiv:2604.21072v2, first posted 2026-04-22, version inspected 2026-05-05. https://arxiv.org/abs/2604.21072

[14] exo-explore contributors. *exo: Run frontier AI locally (repository)*. GitHub, accessed 2026-11-03. https://github.com/exo-explore/exo

[15] Mohammad Shoeybi, Mostofa Patwary, Raul Puri, Patrick LeGresley, Jared Casper, Bryan Catanzaro. *Megatron-LM: Training Multi-Billion Parameter Language Models Using Model Parallelism*. arXiv:1909.08053v4, first posted 2019-09-17, version inspected 2020-03-13. https://arxiv.org/abs/1909.08053

[16] Samyam Rajbhandari, Conglong Li, Zhewei Yao, Minjia Zhang, Reza Yazdani Aminabadi, Ammar Ahmad Awan, Jeff Rasley, Yuxiong He. *DeepSpeed-MoE: Advancing Mixture-of-Experts Inference and Training to Power Next-Generation AI Scale*. arXiv:2201.05596v2, first posted 2022-01-14, version inspected 2022-07-21. https://arxiv.org/abs/2201.05596

[17] Hao Liu, Matei Zaharia, Pieter Abbeel. *Ring Attention with Blockwise Transformers for Near-Infinite Context*. arXiv:2310.01889v4, first posted 2023-10-03, version inspected 2023-11-27. https://arxiv.org/abs/2310.01889

[18] Ruoyu Qin, Zheming Li, Weiran He, Mingxing Zhang, Yongwei Wu, Weimin Zheng, Xinran Xu. *Mooncake: A KVCache-centric Disaggregated Architecture for LLM Serving*. arXiv:2407.00079v5, first posted 2024-06-24, version inspected 2026-10-08. https://arxiv.org/abs/2407.00079

[19] Yuhan Liu, Yihua Cheng, Jiayi Yao, Yuwei An, Xiaokun Chen, Shaoting Feng, Yuyang Huang, Samuel Shen, Rui Zhang, Kuntai Du, Junchen Jiang. *LMCache: An Efficient KV Cache Layer for Enterprise-Scale LLM Inference*. arXiv:2510.09665v2, first posted 2025-10-08, version inspected 2025-12-05. https://arxiv.org/abs/2510.09665

[20] Jiayi Yao, Hanchen Li, Yuhan Liu, Siddhant Ray, Yihua Cheng, Qizheng Zhang, Kuntai Du, Shan Lu, Junchen Jiang. *CacheBlend: Fast Large Language Model Serving for RAG with Cached Knowledge Fusion*. arXiv:2405.16444v3, first posted 2024-05-26, version inspected 2025-04-03. https://arxiv.org/abs/2405.16444

[21] Yuhan Liu, Hanchen Li, Yihua Cheng, Siddhant Ray, Yuyang Huang, Qizheng Zhang, Kuntai Du, Jiayi Yao, Shan Lu, Ganesh Ananthanarayanan, Michael Maire, Henry Hoffmann, Ari Holtzman, Junchen Jiang. *CacheGen: KV Cache Compression and Streaming for Fast Large Language Model Serving*. arXiv:2310.07240v6, first posted 2023-10-11, version inspected 2024-07-19. https://arxiv.org/abs/2310.07240

[22] Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, Ion Stoica. *Efficient Memory Management for Large Language Model Serving with PagedAttention*. arXiv:2309.06180v1, first posted 2023-09-12, version inspected 2023-09-12. https://arxiv.org/abs/2309.06180

[23] Gyeong-In Yu, Joo Seong Jeong, Geon-Woo Kim, Soojeong Kim, Byung-Gon Chun. *Orca: A Distributed Serving System for Transformer-Based Generative Models*. OSDI 2022. https://www.usenix.org/conference/osdi22/presentation/yu

[24] Yinmin Zhong, Shengyu Liu, Junda Chen, Jianbo Hu, Yibo Zhu, Xuanzhe Liu, Xin Jin, Hao Zhang. *DistServe: Disaggregating Prefill and Decoding for Goodput-optimized Large Language Model Serving*. arXiv:2401.09670v3, first posted 2024-01-18, version inspected 2024-06-06. https://arxiv.org/abs/2401.09670

[25] Lianmin Zheng, Liangsheng Yin, Zhiqiang Xie, Chuyue Sun, Jeff Huang, Cody Hao Yu, Shiyi Cao, Christos Kozyrakis, Ion Stoica, Joseph E. Gonzalez, Clark Barrett, Ying Sheng. *SGLang: Efficient Execution of Structured Language Model Programs*. arXiv:2312.07104v2, first posted 2023-12-12, version inspected 2024-06-06. https://arxiv.org/abs/2312.07104

[26] Shi Qiu, Yifan Hu, Xintao Wang, Wenhao Zhu, Jianqin Yan, Hao Chen, Kaiqiang Xu, Kai Chen, Yiming Zhang. *Tutti: Making SSD-Backed KV Cache Practical for Long-Context LLM Serving*. arXiv:2605.03375v1, first posted 2026-05-05, version inspected 2026-05-05. https://arxiv.org/abs/2605.03375

[27] Hui Li, Qihui Zhou, Xiao Yan, Shumeng Zhang, Zijie Dai, Zequn Gong, Mingcong Song, Guoping Long, Hongjie Si, James Cheng. *SARIA: Persistence-Aware KV Caching with Coordinated DRAM--*

SSD Reload for LLM Agents. arXiv:2610.31169v1, first posted 2026-10-22, version inspected 2026-10-22. https://arxiv.org/abs/2610.31169

[28] Yipin Guo, Arun M George, Siddharth Joshi. *GAAC: Lossless KV Compression for Bandwidth-Efficient LLM Inference on Personal Devices*. arXiv:2610.38210v1, first posted 2026-10-28, version inspected 2026-10-28. https://arxiv.org/abs/2610.38210

[29] Yipin Guo, Siddharth Joshi. *SplitZip: Ultra Fast Lossless KV Compression for Disaggregated LLM Serving*. arXiv:2605.01708v3, first posted 2026-05-03, version inspected 2026-06-23. https://arxiv.org/abs/2605.01708

[30] Yann Collet, Murray Kucherawy (Ed.). *Zstandard Compression and the 'application/zstd' Media Type*. RFC 8878, IETF, 2021. https://www.rfc-editor.org/rfc/rfc8878

[31] Abhishek Verma, Luis Pedrosa, Madhukar Korupolu, David Oppenheimer, Eric Tune, John Wilkes. *Large-scale cluster management at Google with Borg*. EuroSys 2015. https://research.google/pubs/large-scale-cluster-management-at-google-with-borg/

[32] Malte Schwarzkopf, Andy Konwinski, Michael Abd-El-Malek, John Wilkes. *Omega: flexible, scalable schedulers for large compute clusters*. EuroSys 2013. https://research.google/pubs/omega-flexible-scalable-schedulers-for-large-compute-clusters/

[33] Mike Burrows. *The Chubby Lock Service for Loosely-Coupled Distributed Systems*. OSDI 2006, pp. 335-350. https://www.usenix.org/legacy/event/osdi06/tech/burrows.html

[34] The Kubernetes Authors. *Leases (Kubernetes documentation, Cluster Architecture)*. kubernetes.io, accessed 2026-11-03. https://kubernetes.io/docs/concepts/architecture/leases/

[35] Murtaza Rangwala, Richard O. Sinnott, Rajkumar Buyya. *Kafila: Serving Large Language Models on a Trusted Set of Heterogeneous Commodity Machines*. arXiv:2609.34045v1, first posted 2026-09-28, version inspected 2026-09-28. https://arxiv.org/abs/2609.34045

[36] Ying Sheng, Lianmin Zheng, Binhang Yuan, Zhuohan Li, Max Ryabinin, Daniel Y. Fu, Zhiqiang Xie, Beidi Chen, Clark Barrett, Joseph E. Gonzalez, Percy Liang, Christopher Ré, Ion Stoica, Ce Zhang. *FlexGen: High-Throughput Generative Inference of Large Language Models with a Single GPU*. arXiv:2303.06865v2, first posted 2023-03-13, version inspected 2023-06-12. https://arxiv.org/abs/2303.06865

[37] Keivan Alizadeh, Iman Mirzadeh, Dmitry Belenko, Karen Khatamifard, Minsik Cho, Carlo C Del Mundo, Mohammad Rastegari, Mehrdad Farajtabar. *LLM in a flash: Efficient Large Language Model Inference with Limited Memory*. arXiv:2312.11514v3, first posted 2023-12-12, version inspected 2024-07-30. https://arxiv.org/abs/2312.11514

[38] Byeongju Kim, Jungwan Lee, Donghyeon Han, Hoi-Jun Yoo, Sangyeob Kim. *FlashMoE: Reducing SSD I/O Bottlenecks via ML-Based Cache Replacement for Mixture-of-Experts Inference on Edge Devices*. arXiv:2601.17063v2, first posted 2026-01-22, version inspected 2026-10-05. https://arxiv.org/abs/2601.17063

[39] Yu Lin, Yiming Wang, Runyuan Cai, Hanze Liu, Xiaodong Zeng. *The Other Half of the Memory Wall: Serving 35B MoEs from SSD with Trained Routing Prediction*. arXiv:2609.18063v2, first posted 2026-09-16, version inspected 2026-09-17. https://arxiv.org/abs/2609.18063

[40] Xijia Wang, TianZhang He, Mohammad Goudarzi. *CEMA-Router: Infrastructure-Aware Batch Routing for LLM Inference in Cloud-Edge Systems*. arXiv:2610.26171v1, first posted 2026-10-19, version inspected 2026-10-19. https://arxiv.org/abs/2610.26171

[41] Tian Wu, Liming Wang, Zijian Wen, Xiaoxi Zhang, Xu Chen, Jingpu Duan, Xianwei Zhang, Jinhang Zuo. *Accelerating Edge Inference for Distributed MoE Models with Latency-Optimized Expert Placement*. arXiv:2508.12851v4, first posted 2025-08-18, version inspected 2026-04-15. https://arxiv.org/abs/2508.12851

[42] Sirui Chen, Jingji Chen, Siqi Zhu, Ziheng Jiang, Yanghua Peng, Xuehai Qian. *Mesh-Attention: A New Communication-Efficient Distributed Attention with Improved Data Locality*.

arXiv:2512.20968v2, first posted 2025-12-24, version inspected 2026-08-09. https://arxiv.org/abs/2512.20968

[43] Yize Wu, Ke Gao, Ling Li, Yanjun Wu. *EasyBalance: Cross-Layer Load Balancing in Distributed MoE Inference*. arXiv:2608.07964v1, first posted 2026-08-08, version inspected 2026-08-08. https://arxiv.org/abs/2608.07964

[44] Pu Li, Jiawen Qi, Qinyu Chen. *When NPUs Are Not Always Faster: A Stage-Level Analysis of Mobile LLM Inference*. arXiv:2605.27435v1, first posted 2026-05-22, version inspected 2026-05-22. https://arxiv.org/abs/2605.27435

[45] Édouard Guégain, Tristan Coignion. *The Battery Price of edge AI: A study of the Environmental Impact of LLM Inference on Mobile Devices*. arXiv:2609.11940v1, first posted 2026-07-10, version inspected 2026-07-10. https://arxiv.org/abs/2609.11940

[46] Juan Parra-Ullauri, Anthony Chiu, Hari Madhukumar, Zhaozhou Wu, Anderson Bravalheri, Alex Healing, Andy Corston-Petrie, Dimitra Simeonidou, Nektaria Efthymiou. *InferSlice: A Reference Architecture for SLO-driven LLM Inference over a Telco Wide-Area Network*. arXiv:2610.32638v1, first posted 2026-10-23, version inspected 2026-10-23. https://arxiv.org/abs/2610.32638

[47] Jason A. Donenfeld. *WireGuard: Next Generation Kernel Network Tunnel*. NDSS 2017. https://www.wireguard.com/papers/wireguard.pdf

[48] Meshtastic project. *Mesh Broadcast Algorithm (Meshtastic documentation)*. meshtastic.org, accessed 2026-11-03. https://meshtastic.org/docs/overview/mesh-algo/

[49] Meshtastic project. *Radio Settings: modem presets (Meshtastic documentation)*. meshtastic.org, accessed 2026-11-03. https://meshtastic.org/docs/overview/radio-settings/

[50] Hugo Krawczyk, Mihir Bellare, Ran Canetti. *HMAC: Keyed-Hashing for Message Authentication*. RFC 2104, IETF, 1997. https://www.rfc-editor.org/rfc/rfc2104

[51] National Institute of Standards and Technology. *Secure Hash Standard (SHS), FIPS PUB 180-4*. NIST, 2015. https://csrc.nist.gov/pubs/fips/180-4/upd1/final

[52] Meshtastic project. *Meshtastic Encryption (Meshtastic documentation)*. meshtastic.org, accessed 2026-11-03. https://meshtastic.org/docs/overview/encryption/

[53] Semtech Corporation. *SX1262 Long Range, Low Power, sub-GHz LoRa RF Transceiver: product page and data sheet (LoRa packet structure, time on air)*. semtech.com, accessed 2026-11-03. https://www.semtech.com/products/wireless-rf/lora-connect/sx1262

[54] Qwen team, Alibaba Cloud. *Qwen/Qwen2.5-0.5B-Instruct model repository (model.safetensors, config.json)*. Hugging Face, accessed 2026-11-03. https://huggingface.co/Qwen/Qwen2.5-0.5B-Instruct

[55] Qwen, :, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tianyi Tang, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, Zihan Qiu. *Qwen2.5 Technical Report*. arXiv:2412.15115v2, first posted 2024-12-19, version inspected 2025-01-03. https://arxiv.org/abs/2412.15115