ARIA: A Peer-to-Peer Efficient AI Inference Protocol

Autonomous Responsible Intelligence Architecture

Anthony MURGO
anthony.murgo@outlook.com

February 2026

Abstract

A purely peer-to-peer AI inference system would allow computational intelligence to be delivered without relying on centralized data centers. We propose a protocol for distributed inference of 1-bit large language models across a network of consumer-grade CPUs. Each node contributes a consented fraction of its compute resources, governed by on-chain smart contracts. The system achieves energy reductions of 70-82% compared to GPU-based inference by leveraging ternary weight architectures (-1, 0, +1). A lightweight blockchain layer provides data provenance, inference verification, and a Proof-of-Useful-Work consensus where mining IS inference. The protocol is model-agnostic and designed for adoption by any existing or future AI system seeking efficient, ethical, and decentralized deployment.

1. Introduction

The current AI infrastructure paradigm relies on massive data centers equipped with thousands of specialized GPUs. Training GPT-3 alone emitted an estimated 552 tons of CO2. Daily inference serving billions of requests multiplies this footprint exponentially. Meanwhile, billions of consumer devices sit idle, their CPUs underutilized.

Recent breakthroughs have shown that Large Language Models can operate with ternary weights (-1, 0, +1) at 1.58 bits per parameter, achieving performance comparable to full-precision models while running efficiently on standard CPUs [1][2]. A 100-billion parameter model can now run on a single CPU at human-readable speed (5-7 tokens/sec) with 82% less energy [3].

We propose ARIA, a protocol that combines three converging innovations: (1) 1-bit model architectures for CPU-native inference, (2) peer-to-peer distribution with explicit consent, and (3) blockchain-based provenance and verification. The result is a system where every connected device can contribute to and benefit from a global AI network, without data centers.

2. The Problem

Existing decentralized AI protocols (Bittensor [4], Gensyn [5], Petals [6]) have advanced the field but share common limitations:

Protocol	Hardware	Focus	Key Limitation
Bittensor	GPU-centric	Training + Inference	Complex, high barrier
Gensyn	GPU	Training	No inference, testnet only
Petals	GPU/CPU	Inference	10-100x latency penalty
DeepSeek	GPU clusters	Full stack	Centralized
ARIA	CPU-first	Inference	This paper

No existing protocol combines: CPU-first architecture, 1-bit model efficiency, explicit consent mechanisms, energy provenance, and blockchain verification in a single coherent system.

3. Protocol Architecture

ARIA operates in three layers: Compute, Consensus, and Service. Each layer is independent and can be adopted separately, but together they form a complete inference protocol.

3.1 Compute Layer

The Compute Layer handles model distribution, inference execution, and resource allocation across the P2P network. It is designed around a fundamental insight: 1-bit models eliminate the need for floating-point multiplication, replacing it with simple addition and subtraction. This makes any CPU a viable inference device.

Model Sharding. A model M of N parameters is partitioned into K shards $S_1...S_K$, where each shard contains a contiguous set of transformer layers. Each node n_i holds one or more shards based on its available memory. For a 2B parameter 1-bit model requiring 0.4GB, even a smartphone can hold the complete model. For larger models, pipeline parallelism distributes layers across nodes.

Inference Pipeline. When a query Q arrives, the orchestrator routes it through the shard pipeline: node n_1 processes layers $L_1...L_k$, passes the intermediate activations to node n_2 for layers $L_{k+1}...L_{2k}$, and so on. The KV-cache is compressed using Multi-head Latent Attention (MLA) [7] to minimize inter-node bandwidth.

Consent Contract. Each node publishes a consent descriptor C_i = {cpu_pct, schedule, task_types, bandwidth_limit, reward_preference}. The orchestrator only routes work to nodes whose consent parameters match the request. Consent can be updated at any time.

```
consent = ARIAConsent(
cpu_percent=25, # Max 25% CPU
schedule='08:00-22:00', # Available hours
task_types=['text_gen'], # Accepted tasks
max_bandwidth_mbps=10 # Network limit
)
```

3.2 Consensus Layer

The Consensus Layer provides three functions: provenance tracking, inference verification, and incentive distribution. It uses a lightweight blockchain optimized for AI metadata.

Proof of Useful Work (PoUW). Unlike traditional PoW that wastes energy on arbitrary computation, ARIA's PoUW rewards actual inference work. A block is produced when a node completes an inference task. The proof includes: hash of the input query, hash of the output, node signature, energy consumed (measured), and latency. Validators verify by re-running a random subset of inferences (spot-checking).

Provenance Ledger. Every inference generates an immutable record R = {query_hash, model_id, node_ids[], output_hash, energy_joules, timestamp, signature}. This enables full traceability of AI outputs to their source data, model, and compute path. The ledger stores only hashes and metadata on-chain; actual data remains off-chain (IPFS/local).

Proof of Sobriety. Each node periodically reports its energy consumption using hardware-level monitoring (RAPL on x86, powermetrics on ARM). These reports are aggregated into a network-wide energy dashboard, providing verifiable proof that the ARIA network consumes less energy per inference than centralized alternatives.

3.3 Service Layer

The Service Layer exposes a standard API compatible with existing AI tooling (OpenAI-compatible endpoints). Any application currently using centralized AI APIs can switch to ARIA by changing a single URL. The layer handles load balancing, request routing, streaming responses, and quality-of-service guarantees.

```
# Drop-in replacement for OpenAI API
from aria import ARIAClient
client = ARIAClient(network='mainnet')
response = client.chat.completions.create(
model='aria-2b-1bit',
messages=[{'role':'user', 'content':'Hello'}]
)
```

4. CPU-Native 1-Bit Inference

The core of ARIA's efficiency comes from native 1-bit inference. Traditional LLMs use 16-bit or 32-bit floating-point weights, requiring expensive multiply-accumulate operations. 1-bit models use ternary weights $W \in \{-1, 0, +1\}$, which transform matrix multiplication into simple additions and subtractions.

For a standard linear layer $y = Wx + b$, where W is the weight matrix and x is the input vector, the 1-bit computation becomes: for each weight w_{ij}, if $w_{ij} = +1$, add x_j; if $w_{ij} = -1$, subtract x_j; if $w_{ij} = 0$, skip. This eliminates all floating-point multiplication, enabling pure integer arithmetic on CPUs.

Metric	Standard (FP16)	1-Bit (ARIA)	Improvement
Memory (2B model)	4.0 GB	0.4 GB	10x less
Energy (x86 CPU)	Baseline	-82%	5.5x less
Speed (x86 CPU)	Baseline	6.17x faster	6x faster
Latency per token	~180ms	~29ms	6x faster
Operations	FP multiply	INT add/sub	No FPU needed

These benchmarks, derived from Microsoft's BitNet research [1][2][3], demonstrate that 1-bit inference on CPU can match or exceed GPU-based inference of standard models for equivalent parameter counts, at a fraction of the energy cost.

5. Peer-to-Peer Network Design

ARIA uses a structured P2P overlay network based on a Kademlia-style DHT (Distributed Hash Table) for node discovery and shard location. Each node maintains a routing table of nearby peers and advertises its capabilities (available shards, compute capacity, consent parameters).

5.1 Node Lifecycle

Join: A new node generates a key pair, downloads model shards from bootstrap peers, publishes its consent contract, and stakes a minimal deposit (anti-Sybil). **Contribute:** The node receives inference requests matched to its shards and consent, processes them, and submits proofs to the ledger. **Earn:** Completed inferences earn ARIA tokens proportional to compute contributed, energy efficiency, and service quality. **Leave:** The node can gracefully disconnect at any time; its shards are automatically redistributed to other peers.

5.2 Fault Tolerance

Model shards are replicated across R nodes (default R=3). If a node becomes unavailable mid-inference, the orchestrator reroutes to a replica within the latency budget. A reputation system tracks node reliability; consistently available nodes earn higher priority in routing.

5.3 Security

Against Sybil attacks: stake-based registration and progressive reputation. Against inference manipulation: statistical verification by cross-checking outputs from multiple independent nodes. Against model poisoning: cryptographic model checksums verified against a pinned registry. Against network attacks: encrypted communications (TLS 1.3), rate limiting, and geographic distribution.

6. Blockchain and Data Provenance

ARIA's blockchain is purpose-built for AI metadata. It does not store model weights or inference data on-chain — only hashes, proofs, and provenance records. This keeps the chain lightweight while providing full verifiability.

6.1 On-Chain Records

```
InferenceRecord {
id: bytes32 // Unique record hash
query_hash: bytes32 // SHA-256 of input
output_hash: bytes32 // SHA-256 of output
model_id: bytes32 // Model version hash
nodes: address[] // Contributing nodes
energy_mj: uint64 // Millijoules consumed
latency_ms: uint32 // Total latency
timestamp: uint64 // Unix timestamp
signature: bytes // Aggregated signature
}
```

This structure enables: (a) verification that an output was genuinely produced by the claimed model on the claimed nodes, (b) energy auditing at the per-inference level, (c) tracing any AI output back to its complete provenance chain, and (d) dispute resolution through cryptographic evidence.

6.2 Smart Contracts

Four contract types govern the protocol: **ConsentRegistry** stores and enforces node consent parameters; **InferenceMarket** matches requests to nodes and manages payments; **ProvenanceLedger** records immutable inference attestations; **RewardDistributor** calculates and distributes token rewards based on contribution metrics.

7. Incentive Design

ARIA's incentive mechanism is designed to be simple and fair. Nodes earn tokens for useful work. The reward formula balances quantity, quality, and efficiency:

Reward(n) = base_rate × inferences_completed × quality_score × efficiency_bonus

Where **quality_score** = f(uptime, latency, verification_pass_rate) ∈ [0, 1], and **efficiency_bonus** = g(energy_per_inference / network_average) ∈ [0.5, 2.0]. Nodes that consume less energy per inference than the network average earn up to 2x bonus. This directly incentivizes energy efficiency and rewards CPU nodes over GPU nodes for equivalent work.

The token supply follows a halving schedule inspired by Bitcoin: initial emission is reduced by 50% every 4 years, creating predictable scarcity. Unlike Bitcoin, the work performed during mining (inference) is inherently useful, eliminating the energy waste criticism of PoW systems.

8. Reference Implementation

A reference implementation accompanies this paper, written in Python for accessibility. It demonstrates the core protocol mechanics: P2P node discovery, consent-based inference routing, model sharding, provenance recording, and energy monitoring. The implementation is intentionally minimal (~800 lines) to serve as a starting point for the community.

Key components:

File	Purpose	Lines
node.py	Core ARIA node: join, contribute, earn	~200
network.py	P2P networking, discovery, routing	~150
inference.py	Distributed 1-bit inference engine	~150
ledger.py	Provenance blockchain / ledger	~150
consent.py	Consent contracts and enforcement	~80
proof.py	PoUW and Proof of Sobriety	~80

The reference implementation runs on any machine with Python 3.10+. No GPU is required. A demonstration network can be launched locally with 3+ nodes to observe the full protocol cycle: discovery, consent, inference, provenance, and rewards.

```
# Launch an ARIA node in 3 lines
from aria import ARIANode
node = ARIANode(cpu_percent=25, port=8765)
node.run() # Join network, start contributing
```

9. Future Work and Invitation

This paper and reference implementation describe the foundation of the ARIA protocol. Significant work remains, and we explicitly invite the community to extend, improve, and challenge this design:

Model scaling: Extend the sharding protocol to efficiently distribute models with 100B+ parameters across thousands of heterogeneous nodes. **Multi-modality:** Adapt the inference pipeline for image, audio, and video models. **Formal verification:** Prove security properties of the consent and verification mechanisms. **Economic modeling:** Simulate tokenomics under various adoption scenarios. **Hardware optimization:** Develop specialized 1-bit kernels for RISC-V, NPUs, and DSPs. **Privacy:** Integrate zero-knowledge proofs for private inference without revealing inputs.

ARIA is released under the MIT License. The protocol belongs to no one and to everyone. We believe that AI infrastructure should be a public good — like the internet itself — and that the path to ethical AI runs through radical transparency, distributed ownership, and technological sobriety.

10. Conclusion

We have presented ARIA, a protocol for peer-to-peer AI inference that combines 1-bit model efficiency, CPU-first computing, blockchain provenance, and explicit consent. The protocol addresses the fundamental tension in modern AI: the need for powerful intelligence versus the cost of centralized infrastructure.

By making every connected device a potential contributor to a global AI network, ARIA offers a path toward AI that is efficient (70-82% less energy), accessible (no GPU required), transparent (full provenance), ethical (explicit consent), and open (MIT licensed). The reference implementation demonstrates that this is not a theoretical exercise but a practical architecture ready for community adoption and improvement.

The era of centralized AI infrastructure need not be permanent. Just as BitTorrent decentralized file sharing and Bitcoin decentralized money, ARIA proposes to decentralize intelligence itself.

References

[1] Ma, S. et al. "The Era of 1-bit LLMs: All Large Language Models are in 1.58 Bits." arXiv:2402.17764, 2024.
[2] Wang, H. et al. "BitNet: Scaling 1-bit Transformers for Large Language Models." arXiv:2310.11453, 2023.
[3] Microsoft Research. "bitnet.cpp: Efficient Edge Inference for Ternary LLMs." GitHub, 2025.
[4] Bittensor Foundation. "Bittensor: A Peer-to-Peer Machine Intelligence Network." Whitepaper, 2021.
[5] Gensyn. "A Protocol for Deep Learning Computation." Technical Report, 2023.
[6] Borzunov, A. et al. "Petals: Collaborative Inference and Fine-tuning of Large Models." NeurIPS Workshop, 2022.
[7] DeepSeek AI. "DeepSeek-V3 Technical Report." arXiv, 2024.
[8] DeepSeek AI. "DeepSeek-R1: Incentivizing Reasoning Capability in LLMs." arXiv:2501.12948, 2025.
[9] Nakamoto, S. "Bitcoin: A Peer-to-Peer Electronic Cash System." 2008.
[10] Microsoft. "BitNet b1.58 2B4T Technical Report." Hugging Face, 2025.