Infernet Protocol: A Peer-to-Peer GPU Inference Marketplace

Technical Whitepaper

Infernet Protocol

2026

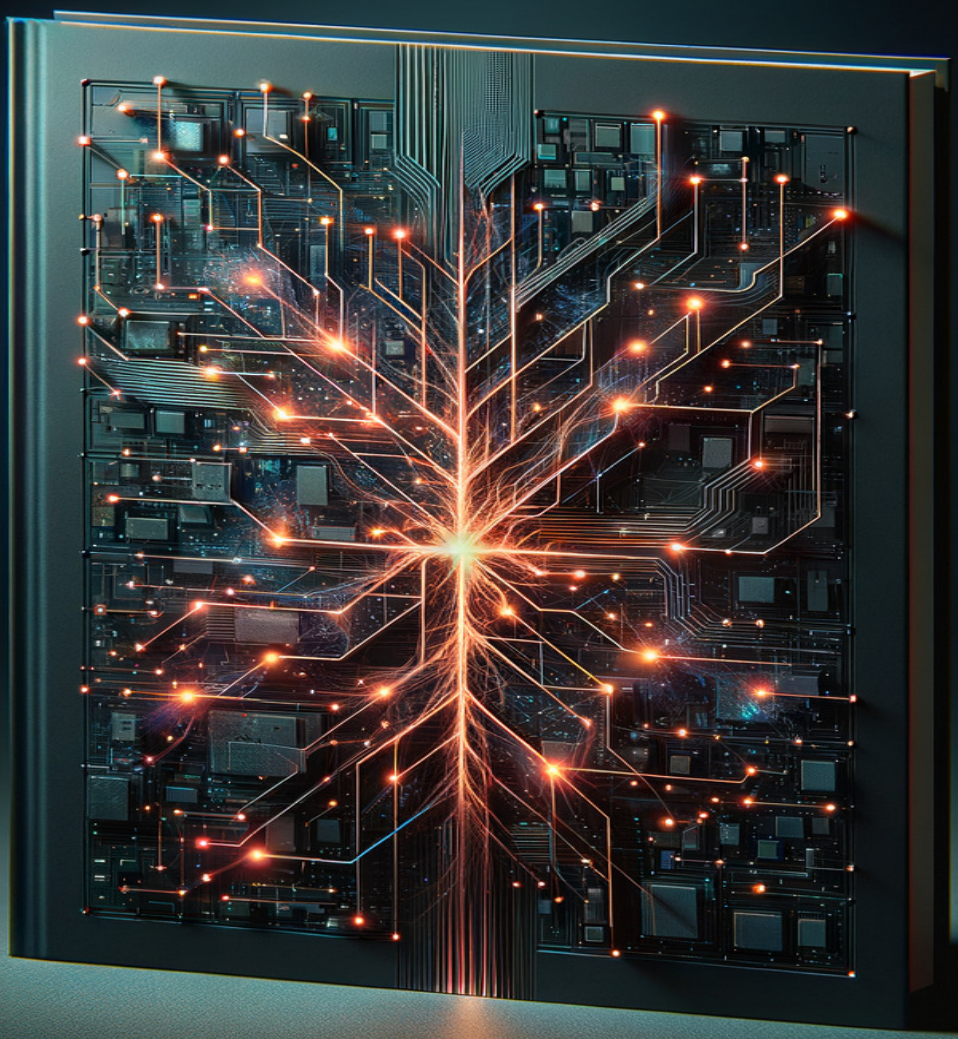
Abstract

Infernet Protocol is an open, permissionless marketplace for GPU inference. Operators install a single CLI tool, register their hardware, and immediately begin earning cryptocurrency for every inference job their node completes. Consumers pay per request in any supported coin with no account required. The network has no native token, no rent extraction, and no central coordinator — operators and consumers interact directly through a Nostr-signed HTTP protocol built on standard OpenAI-compatible endpoints. This paper describes the network's architecture, economic model, security properties, and roadmap.

Infernet Protocol

The Infernet Protocol Book

Open-Source Guide to Decentralized GPU Inference



Infernet Protocol

A Peer-to-Peer GPU Inference Marketplace

Technical Whitepaper

infernetprotocol.com

github.com/infernetprotocol/infernet-protocol

Contents

1 Introduction

The commoditization of large language models has outpaced the infrastructure for distributing their compute. Today's GPU inference market is dominated by a handful of cloud hyperscalers and purpose-built inference startups, all of which extract a percentage of every API call, require account creation, and impose rate limits or geographic restrictions.

Infernet Protocol eliminates these intermediaries. A node operator with an NVIDIA or AMD GPU runs one command:

```
curl -fsSL https://infernetprotocol.com/install.sh | bash
```

Within minutes the node is registered, its hardware fingerprinted, and it begins bidding on inference jobs. Consumers send standard OpenAI-compatible HTTP requests to the network endpoint; the protocol routes each request to a healthy provider node, returns the response, and records the transaction. Payment settles in the coin chosen by the consumer; the operator's payout address is on-chain from day one.

No native token is issued. No DAO governance is required. Fees are charged in the same coin the consumer pays — the protocol takes a small platform fee at settlement and passes the remainder directly to the operator.

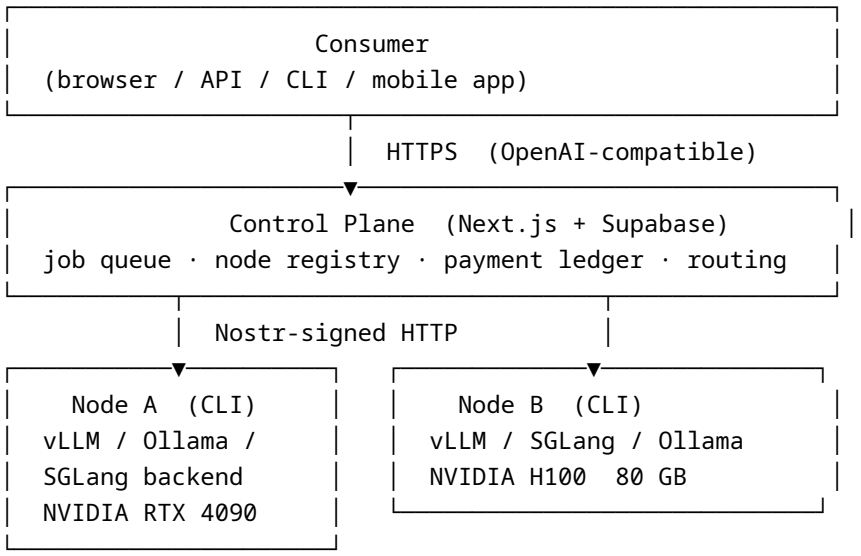
1.1 Goals

1. **Zero-friction onboarding** — a GPU node should be earning in under five minutes on any Linux server.
2. **Neutral payment rails** — consumers pay in BTC, ETH, SOL, USDC, USDT, or a dozen other coins; operators set their preferred payout coin.
3. **No rent extraction** — the protocol's fee is fixed and transparent; operators keep the majority of every payment.
4. **Open and auditable** — the full stack (CLI, daemon, control plane, payment integration) is MIT-licensed on GitHub.
5. **Hardware-agnostic** — NVIDIA, AMD, and Apple Silicon nodes are all first-class citizens.

---

2 Architecture Overview

Infernet Protocol has three layers: the **control plane**, the **P2P network**, and the **inference engine**.



2.1 Control Plane

The control plane is a Next.js application backed by Supabase (Postgres). It is the only centralized component, and its role is intentionally narrow:

- **Node registry** — stores signed node records (pubkey, GPU specs, model list, endpoint, VRAM tier).
- **Job queue** — matches incoming consumer requests to available nodes.
- **Payment ledger** — records payment intents, on-chain confirmations, and payout instructions.
- **Routing** — returns a healthy, capable node to the consumer for each request; consumers then talk directly to that node.

The control plane never holds private keys, never proxies inference traffic, and never stores model weights. Operators may self-host the entire control plane using the open-source repository.

2.2 Node Daemon

Each operator node runs the `infernet` CLI as a background daemon. The daemon:

1. **Registers** the node by POSTing a Nostr-signed record to the control plane.
2. **Heartbeats** every 30 seconds with current load and availability.
3. **Accepts jobs** forwarded by the control plane and proxies them to the local inference engine (Ollama, vLLM, or SGLang).
4. **Reports completion** with token counts and latency metrics.
5. **Auto-upgrades** by checking the npm registry for new CLI versions and re-registering after update.

The daemon stores no secrets beyond the operator's Nostr private key, which never leaves the node.

2.3 Nostr Identity and Request Signing

Node authentication uses Nostr secp256k1 keypairs. Every node registration, heartbeat, and status update is signed with the node's private key and verified by the control plane against the public key stored in the registry.

This design has two properties we consider important:

- **No password database** — the control plane stores public keys only. Compromise of the control plane's database does not expose operator credentials.
- **Self-sovereign identity** — an operator can migrate their node to a new server without contacting the platform; the keypair travels with them.

CLI device-code login for the web dashboard uses a separate short-lived JWT derived from a server-side HMAC secret and the operator's verified email.

2.4 Inference Engines

The daemon supports three local backends:

Backend	Best for	Notes
Ollama	General-purpose LLMs up to ~70B	Easiest setup; GGUF quantized
vLLM	High-throughput, full-precision HuggingFace models	Requires CUDA; PagedAttention
SGLang	Structured generation, long-context models	Supports Llama, Mistral, Qwen

All three expose an OpenAI-compatible `/v1/chat/completions` endpoint. The daemon auto-detects whichever is running on standard ports (11434 for Ollama, 8000 for vLLM, 30000 for SGLang).

Operators can also download HuggingFace models directly:

```
infernet model pull hf:NousResearch/Hermes-3-Llama-3.1-8B
infernet uncensored   # auto-picks best uncensored model for available VRAM
```

---

3 Hardware Classification

Nodes self-report their hardware at registration time. The daemon uses `nvidia-smi`, `rocm-smi`, and `/proc/meminfo` to collect:

- GPU model, VRAM (MB), architecture generation
- CPU core count, RAM (MB)
- Operating system and CUDA version

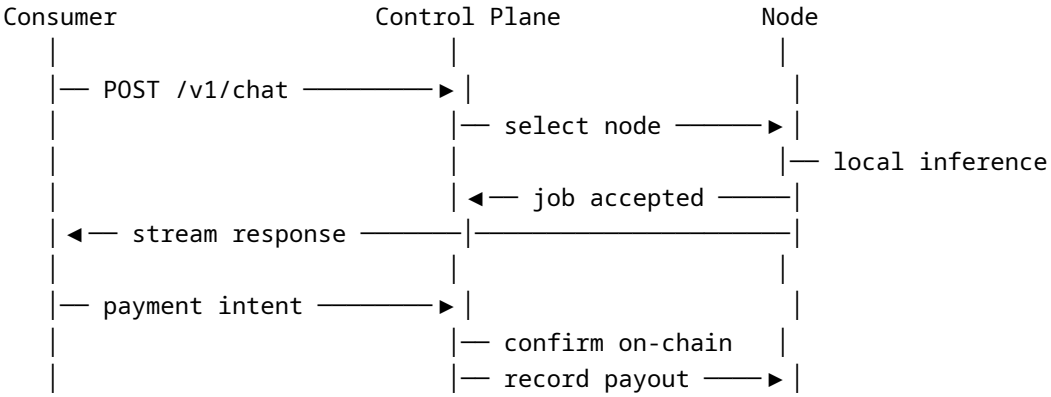
VRAM is bucketed into tiers for routing:

Tier	VRAM	Typical hardware
`nano`	< 4 GB	Integrated / low-end mobile GPUs
`small`	4–8 GB	RTX 3060, RTX 4060
`medium`	8–16 GB	RTX 3080, RTX 4080
`large`	16–32 GB	RTX 3090, RTX 4090, A100 40 GB
`xl`	32–80 GB	A100 80 GB, H100 80 GB
`xxl`	80 GB+	Multi-GPU, H100 NVLink clusters

The routing layer uses these tiers to filter nodes before matching a job. Operators on shared hosting platforms (Vast.ai, RunPod) are supported natively; the daemon automatically excludes Docker bridge IPs from its advertised endpoint.

---

4 Job Lifecycle



1. **Consumer submits** a chat completion or embedding request to the control plane's OpenAI-compatible endpoint.
2. **Routing** selects a healthy node with sufficient VRAM tier and the requested model. Selection prefers lower latency and higher completion rate.
3. **Node executes** the inference locally and streams the response back.
4. **Metering** records token counts (prompt + completion) and wall-clock latency for billing.
5. **Payment** is settled asynchronously: the consumer's payment intent triggers a CoinPayPortal transaction, which webhooks back to the control plane on confirmation.
6. **Payout** accumulates in the operator's ledger and can be withdrawn at any time to any address they configure via `infernet payout set`.

---

5 Payment Model

5.1 No Native Token

Infernet Protocol deliberately issues no native token. Token issuance creates misaligned incentives: the platform benefits from token price appreciation regardless of actual inference utility delivered. Instead, operators and consumers transact in established coins with deep liquidity.

5.2 Supported Coins

Payment is processed via CoinPayPortal, which supports:

- **Bitcoin** (BTC) — on-chain and Lightning Network
- **Ethereum** (ETH) and ERC-20 stablecoins (USDC, USDT)
- **Solana** (SOL) and SPL tokens
- **Polygon**, **BNB Chain**, **XRP**, **ADA**, **DOGE**
- Multi-chain USDC and USDT

Operators configure a payout address per coin:

```
infernet payout set --coin BTC --address bc1q...
```

The platform converts consumer payments to operator payouts at the prevailing exchange rate with no spread markup — the platform fee is the only deduction.

5.3 Fee Structure

Party	Amount
Consumer	Pays per 1 000 tokens (prompt + completion)
Operator	Receives ~90% of consumer payment
Platform	Retains ~10% as protocol fee

Exact per-token rates are set by operators within bounds published by the control plane. Operators competing for the same job tier drive rates toward the market equilibrium.

5.4 Reputation (CPR)

Completed jobs generate a signed receipt issued to CoinPay's Reputation Protocol (CPR). Operators accumulate on-chain reputation scores that affect their routing priority. High-reputation nodes receive more job offers; nodes with repeated failures or timeouts are deprioritized automatically.

---

6 Security Model

6.1 Control Plane Compromise

If the control plane is compromised, an attacker gains access to:

- The node registry (public keys and endpoint URLs — both already public)
- The job queue and payment ledger (financial metadata, no private keys)
- Consumer email addresses (Supabase Auth)

An attacker does **not** gain:

- Operator Nostr private keys (stored on-node only)
- Model weights (stored on-node only)
- Consumer payment credentials (handled by CoinPayPortal)

6.2 Node Impersonation

A malicious actor cannot register a fake node under an existing operator's identity without possessing the operator's Nostr private key. All node mutations (register, heartbeat, update) must be signed; the control plane rejects unsigned or incorrectly signed requests.

6.3 Job Integrity

The control plane does not guarantee inference result integrity in v1. Output verification (challenge-response or ZK proof of execution) is on the roadmap (see §8). Operators with sustained low accuracy or high refusal rates are penalized in routing.

6.4 Model Key Hierarchy (IPIP-0028)

Each model artifact published to the network is assigned a secp256k1 keypair. This enables three-party encrypted payloads between consumer, node, and model — ensuring that sensitive prompts can be encrypted end-to-end with NIP-44 without the control plane being able to read them. IPIP-0028 is implemented in the CLI's `model-key.js` module and will be extended as the protocol matures.

---

7 Decentralization Roadmap

Infernet Protocol ships today as a semi-decentralized system: the P2P network and payment rails are decentralized, but the job queue and node registry live on a centralized control plane. The phased roadmap moves progressively toward full decentralization:

Phase	Description	Status
0	Centralized control plane (Supabase + Next.js)	**Shipped**
1	Nostr-signed node identity; no password database	**Shipped**
2	Multi-coin payments via CoinPayPortal	**In progress**
3	Reputation protocol (CPR receipts per job)	**Planned**
4	Decentralized job queue (Nostr relay or DHT)	**Planned**
5	ZK proof of inference execution	**Research**
6	Fully P2P — no required central coordinator	**Research**

The centralized control plane in Phase 0 is intentional. Eliminating it before the economic incentives are proven would sacrifice reliability without a corresponding user benefit. Each subsequent phase reduces trust assumptions while preserving the operator and consumer experience.

---

8 Distributed Training

Beyond inference, Infernet Protocol is designed to support distributed model training across the operator network. The `infernet train` command (in active development) will allow:

- **Data collection** — operators contribute curated datasets to training runs.
- **Federated fine-tuning** — gradient updates aggregated across nodes without sharing raw training data.
- **Model publishing** — completed models pushed to the InfernetProtocol HuggingFace organization for public access.

Training jobs will use the same payment model as inference: consumers pay per GPU-hour, operators earn per gradient step contributed.

---

9 Comparison

Infernet Protocol	Centralized API (OpenAI et al.)	Other Decentralized Networks
No native token required	No	Often yes
Yes	No	Varies
Operator self-custody		

Infernet Protocol	Centralized API (OpenAI et al.)	Other Decentralized Networks
OpenAI-compatible API	Yes	Yes	Rarely
Multi-coin payments	Yes	No (card / credits)	Single chain
Open source	MIT	No	Varies
Setup time	< 5 min	Account + billing	Hours–days
GPU hardware agnostic	Yes	N/A	Often NVIDIA only

---

10 Getting Started

As an operator:

```
# Install
curl -fsSL https://infernetprotocol.com/install.sh | bash

# Configure
infernet init

# Register and start earning
infernet register
infernet start
```

As a consumer:

Send standard OpenAI-compatible requests to `https://infernetprotocol.com/v1/` with your API key. No account required for the public inference endpoint.

```
from openai import OpenAI

client = OpenAI(
    base_url="https://infernetprotocol.com/v1",
    api_key="your-api-key"
)

response = client.chat.completions.create(
```

```
    model="qwen2.5:7b",
    messages=[{"role": "user", "content": "Hello!"}]
)
print(response.choices[0].message.content)
```

---

11 Conclusion

Infernet Protocol demonstrates that a GPU inference marketplace can operate without a native token, without a centralized owner, and without compromising the operator or consumer experience. By grounding identity in Nostr keypairs, payments in established coins, and inference in standard OpenAI-compatible APIs, the protocol slots into existing workflows while laying the foundation for a fully decentralized compute layer.

The source code is MIT-licensed at github.com/infernetprotocol/infernet-protocol. Operators, consumers, and contributors are welcome.

References

1. Nakamoto, S. (2008). Bitcoin: A Peer-to-Peer Electronic Cash System.
2. Fiatjaf et al. (2020). Nostr Protocol. https://nostr.com
3. Kwon, J., Buchman, E. (2019). Cosmos: A Network of Distributed Ledgers.
4. vLLM Project. PagedAttention: Efficient Memory Management for LLM Serving. https://vllm.ai
5. Ollama. Run Large Language Models Locally. https://ollama.com
6. CoinPayPortal. Multi-Currency Payment Rails. https://coinpayportal.com
7. Hugging Face. The AI Community Building the Future. https://huggingface.co