DecenMind: A Peer-to-Peer Artificial Intelligence Network

Jensen Wang*

Genesis draft v0.4 · 28 November 2026

* A pseudonym. No affiliation with NVIDIA or any other company.

Draft, published to be attacked. Part of this design now runs: a coordinator is live on a test chain that only the project's own listed test nodes can join, with one reference job. Its ledger is public (read it live) and it will be reset. No AI model runs on it yet. DCM, the network's coin, is not minted yet (its rules), and the project sells none. What earlier drafts got wrong is listed in section 13.

Companions: technical architecture · economics and simulation · this paper as PDF

ABSTRACT

A purely peer-to-peer AI network would let anyone obtain machine intelligence by contributing compute instead of paying an operator. Idle consumer graphics cards can already run useful open models, and volunteer networks have shown that strangers will serve each other's requests. What those networks lack is a way to pay for work without trusting a coordinator, and a way to know the work was done. We propose a network in which requests are escrowed, matched peer to peer, executed in a sandbox against content-addressed models, audited at random by bonded verifiers, and settled automatically. DCM, the network's unit, is made only by verified work: a fixed amount each day, shared by the work verified that day. Machines are meant to add up, so that many small ones run what no single one can. The design target is strict: the network must keep working after its founder switches off every machine he controls. This draft states what is known, what is assumed and what is unsolved.

1 Introduction
2 Goals and non-goals
3 Participants
4 Models as content
5 Life of a job
6 Proof of useful compute
7 Economics
8 Network
9 Gateways
10 Safety
11 Founder independence
12 Open problems
13 Changes
References

1 Introduction

Access to AI is rented from a small number of operators who own the hardware, set the price and decide who may use it. Meanwhile most consumer graphics cards sit idle most of the day. The wish to close that gap is not new. A market for strangers' computers was proposed in 2016[2], and replacing wasteful mining with machine-learning work was proposed in 2019 and 2020[3,4].

Three things have changed since. Open models that are good enough to be useful now fit on a single gaming card. Volunteer networks have served image and text generation on donated hardware for years[5], and layered inference of large language models across volunteers' machines has been demonstrated[6]. And a 72-billion-parameter language model has been trained from scratch by permissionless participants coordinated over a public chain[7].

What has not been shown is a network of this kind that pays its operators from real demand instead of subsidy, that detects fake work cheaply on mixed consumer hardware, and that survives the removal of whoever started it. DecenMind is an attempt at those three.

2 Goals and non-goals

Goals:

1. Anyone can join with no permission, with one program or from a browser.
2. Machines add up: together they do what no single one can, in the number of jobs and in the size of a model (section 5.1).
3. Work is rewarded only when it has been verified.
4. A participant who gives more compute than they use never pays.
5. No server, domain or key held by the founder is required for operation.
6. The price of AI is quoted in units of work, so it does not swing with the price of a coin.

Non-goals, stated so that nobody is misled:

- **Training frontier models.** The largest permissionless training run to date required data-centre hardware from every participant[7]. Gaming PCs on home connections are suited to inference.
- **Hiding a request from the machine that runs it.** Consumer cards have no trusted execution environment.
- **Beating data centres on latency.** The first workloads are the ones that tolerate a queue: images, 3D assets, video, speech, embeddings, and text where cost matters more than the first second.
- **Making node owners rich.** A gaming card earns little (section 7.5). The offer is free AI first and income second.

3 Participants

An identity is a key pair. One program can hold several roles at once, according to the hardware it runs on.

ROLE	DOES	BRINGS
Requester	Signs jobs and backs them with credit or DCM	Demand
Worker	Runs models on requests	GPU, memory

Verifier	Re-runs a random share of jobs	GPU, a bond
Seeder	Stores and serves model files	Disk, uplink
Relay	Connects peers that cannot reach each other	Public address, bandwidth
Gateway	Offers a website or API to people without a node	A front door

4 Models as content

Only models listed in a public registry are run. An entry is the hash of the weights, the hash of the runtime that executes them, the licence, and a reference benchmark: how many work units one job costs on reference hardware. A job is a tuple of model, input, seed and parameters. A worker never executes code supplied by a requester.

Some widely used open models carry licences that forbid commercial serving or exclude whole territories. Those models are not listed. The registry begins with permissively licensed models only.

5 Life of a job

1. **Request.** The requester's client signs the job and the most it will give for it, in credit or in DCM.
2. **Escrow.** That amount is locked before any work starts.
3. **Match.** The offer is gossiped. Workers that already hold the model answer. For paid jobs the client chooses by price, bond and history. For credit-backed and unbacked jobs the worker is drawn at random, never chosen (section 7.2). The design has no central scheduler; today the project's one coordinator does the matching (section 9).
4. **Execute.** The worker runs the job in a contained process, with limits on its memory, processor, processes and lifetime. A full sandbox is not built yet (section 10).
5. **Commit.** The worker publishes a commitment (section 6.1) and hands over the output. By design it goes straight to the requester; today it waits one day in the

project's file storage.

6. **Audit window.** The job may be selected for re-execution.
7. **Settle.** Escrow is released to everyone who had a part in the job (section 7.3).

A worker may give a job back. If it is busy, the job keeps its place. If the job failed on it, the job is never drawn for that worker again, and it is refused once it has failed on three workers or on every worker that could take it. A worker that gave a job back is left out of the draw for a short while, so that one job is never passed back and forth.

5.1 Adding up

The point of a network is that machines add up. Today one job runs on one machine, so the network adds up in jobs: a hundred cards finish a hundred jobs at once. The design also adds up in size, in three ways. **In a line:** each machine holds consecutive blocks of one model and the work passes down the line. Others have shown this for large language models over the internet[6]; it gives size, not speed, and it is neither built nor measured here. **By stages:** different machines do different stages of one job, a shape on one and its texture on another. **By pieces:** a large job is cut into independent parts, done at once and joined. Stages and pieces need no new trust. A line needs audits of a slice, direct links between machines, and receipts that pay each machine on it.

If a worker disappears, the job times out and returns to the market. The escrow is untouched. A job that has not settled within a day of its offer expires, so that no node has to remember every job for ever.

6 Proof of useful compute

Bitcoin's work is trivial to check: a hash meets the target or it does not[1]. AI work is not. For the same model and seed, two honest machines produce outputs that differ in the last bits, because floating-point results depend on hardware, driver and precision. Comparing hashes of outputs therefore proves nothing.

There are three known routes around this. Cryptographic proofs of inference remain orders of magnitude more expensive than the inference itself[8]. Trusted hardware is absent from consumer cards. Bitwise-reproducible operators exist, at a performance cost and for a limited set of models so far[8]. For the first version we take a fourth, economic route, and treat the others as later upgrades.

6.1 Commit

Before returning the output, the worker publishes a commitment: the hash of the output, so that the requester can prove what was delivered, and a compact fingerprint of intermediate state. For language models the fingerprint is a locality-sensitive hash of activations[9]. For diffusion models it is the latent at fixed steps, reduced in size.

6.2 Audit

Each job is selected for audit with probability p. The selection is the hash of the commitment combined with a public beacon value that did not exist when the worker committed, so it cannot be predicted or steered. The same value draws the verifier from the bonded set, never the worker or the requester. The beacon is a public randomness service run by independent operators[11]. The round a job uses follows from the time written in its commitment and lies a few rounds in the future, so neither the worker nor the coordinator can choose it, and anyone can fetch the same round and check. Later it may be a block hash of the settlement chain.

The verifier re-runs the job with the same seed and compares fingerprints within a tolerance calibrated per model across hardware.

6.3 Bond

Every worker holds a bond B that is destroyed on a failed audit. Let g be the most a worker can gain by faking one job, and d the chance that an audit catches a fake. Faking is unprofitable when

$$p \cdot d \cdot B > g$$

With $p = 2\%$ and a perfect audit, a bond worth fifty jobs is just enough. An audit that catches only half of all fakes needs a bond of a hundred. Honest participants pay only the audit overhead, which is p.

The bond is measured against the job: fifty jobs' worth of the job being done, so a larger job needs a larger bond. It is held in credit, and paid jobs add a bond in DCM. The starting values sit at the edge of the inequality and will be reset from each model's measured catch rate.

6.4 A bond without buying in

A new worker has no bond and should not have to buy one. Their first earnings are withheld until the bond is full. Until then they are audited at a higher rate and nothing they earn can be spent. Joining is free. Cheating never is. A thousand fresh identities hold a thousand empty bonds, which blunts the cheapest Sybil attack.

6.5 Verifiers

Verifiers are bonded too. A verifier is paid for every audit the draw requires, and paid the same whatever its verdict, so that no verdict pays better than another. To stop a verifier from approving everything without doing the work, the network mixes in jobs whose correct fingerprint is already known. A dispute escalates to more verifiers and the losing side is slashed.

6.6 The requester's audit

A requester who doubts a result can force an audit by posting its cost. If the audit fails, the cost is returned with a share of the slashed bond. If it passes, the verifier keeps it. Nobody can dispute a result on taste. Only re-execution decides.

6.7 What an audit reveals

An audited job's input is shown to the verifier. A requester should assume that any request may be read by two machines, not one.

6.8 Limits

Tolerance invites cheating inside the tolerance: a smaller quantisation, fewer steps, a cheaper model that looks similar. Fingerprints must be chosen so that these shortcuts fall outside it. That is demonstrated for some language models and unproven for most image, video and 3D models. It is the main research task of this project, and until it is done the guarantee is economic, not cryptographic.

7 Economics

Work is counted in units and paid for in DCM. There is no other way to pay: a wallet address is the only identity in the network, and there are no accounts, no cards and no ordinary money anywhere in it. Numbers and a simulation are in the companion page.

7.1 Work units

Prices are listed in work units. One unit is one second of a reference card on a reference job, and each registry entry states its cost in units.

7.2 The ledger today: credit

Today the ledger counts in units, called credit. Credit goes to a worker when a job it served settles, one unit per unit of verified work, and a requester spends it to give a job priority. It cannot be transferred, and it decays, so that old claims do not pile up against today's capacity. A worker's first credit fills its bond (section 6.4), and the bond cannot be spent. This is a test chain: it will be reset, and its credit is not the coin of section 7.8.

Requests are served in order of what backs them. A request with nothing behind it is served last, from capacity that would otherwise be idle, and the worker that serves it still earns credit. That is where the first credit comes from: doing work, for anyone. The longest-running volunteer network has operated on this principle for years[5].

Unbacked and credit-backed requests are assigned by a random draw, not chosen by the requester. A participant therefore cannot route its own requests to its own machine

and mint credit for work that served nobody. Nor can it hand its credit to someone else through a job that was never really done. Most jobs are never audited, so without the draw such a job would almost always pass, and credit could be sold.

Units per job without a ceiling are safe only while the door is shut. Once anyone may join, anyone could send the network jobs only to earn units. So the coin does not follow units one for one: a fixed amount comes out each day and is shared by that day's verified work (section 7.8).

7.3 Paying

A requester who wants priority without having contributed pays, in DCM, from their own wallet. The payment is locked for that job and split at settlement. The proposed split: the worker 70%, the gateway that brought the job 10%, the verifier 5%, the model's owner 5%, relays 5%, and 5% burned. There is no treasury: the project earns from a paid job only what its own machines and its own gateway earn. The library still carries an earlier split with a treasury, and the paid lane is not open. Paid work is served before credit work, and a worker that serves it earns the payment and the credit. This is the only payment that enters the system.

7.4 Contribute to create

A person whose machine serves 100 units and who uses 80 never pays. A person who uses more than their machine gives waits in the unbacked queue for the difference, or pays for it.

7.5 Free entry

A worker stays only while what it earns exceeds its electricity. Nobody needs permission to join, so people join until the typical operator nets a few dollars a month, whatever the demand. More demand brings more operators, not richer ones. In the companion model an 8 GB card nets about eight cents for an hour of paid work and settles near a tenth of its hours being paid. The size of the network follows demand. This is intended.

7.6 No pay for uptime, no coins before work

Being online can be faked for nothing; verified work cannot. A network that pays for uptime issues claims on compute that nobody has agreed to supply. A network that hands out an asset for being online pays out exactly what speculative buyers put in: the companion model grows such a network to more than thirty times the size its demand supports, with almost all machines waiting, and then shrinks it at a loss to the last people who joined. The coin of section 7.8 is built against that result: none is made before work, and each day's amount goes only to verified work, never to hours online. The companion model does not cover the schedule yet.

7.7 Self-dealing

A participant who pays for jobs from themself loses the fee each time. A participant who backs a job with credit, or with nothing, cannot choose who serves it (section 7.2), and a worker never serves its own request. The draw closes the credit and free lanes, and the burn closes the paid one: a group that pays itself loses the burned share and whatever the draw hands to other workers, and gains nothing that an unpaid job would not have given it. That is also why the daily amount does not depend on paid work (section 7.8).

7.8 The coin

DCM is the network's coin, on Solana, capped at 21 million. It is not minted yet. Its rules have a page of their own; in short: a schedule gives 7,186.86 a day for the first four years, 10.5 million in all, and halves every four years. Each day's coins are shared by the verified work of that day, in proportion; a verifier's required audit counts as work. All of each day's amount comes out, every day, as Bitcoin's does; nothing opens or closes it. An interim draft held four fifths back until paid work grew, and a model showed that a group paying itself could open that gate at a profit.

One fifth of the first 10.5 million coins goes to development: at most 2.1 million, a tenth of all there will ever be, written into the program that mints, and after that nothing. No coin is made before work. The project sells none. The founder mines with his own machines like anyone else.

The mint is a program, not a person. While the network is tested in the open the program can still be changed, and this paper will say so for as long as that is true; the right to change it is given up at the end (section 11). The price of AI is quoted in work units, so it does not follow the coin's price, and the network must work if no exchange ever lists the coin.

Paid jobs will settle on the same chain. Because jobs are small, a requester opens a channel with one deposit and signs a receipt per job; workers claim receipts in batches; only disputes go to the chain one at a time.

8 Network

Peers find each other through a distributed hash table seeded from several independent lists, which anyone may publish. Most home machines sit behind address translation. Hole punching in a large deployed peer-to-peer network has been measured at about 70% success[10], so relays are a paid role in the protocol, not a service the founder runs.

By design job offers travel by gossip, and inputs, outputs and models travel directly between peers; models are large, so they are distributed by swarming, and the machines that carry data share the relays' part of a paid job. None of that is built yet: today offers go through the coordinator, and files wait one day in the project's storage.

9 Gateways, and what is central today

A website or API is an ordinary client that speaks the protocol on behalf of people who do not run a node. No gateway is built yet. Anyone will be able to run one, and the proposed split gives a gateway a tenth of the paid jobs it brings. A gateway is a front end only: its users connect their own wallets and pay in DCM themselves, and it never holds their funds.

The network today is centralised, and this paper says so plainly: one coordinator, one verifier, the model list, file storage and the site are the project's. The coordinator can refuse, delay or drop a job. It cannot forge a message, change a balance, skip a required

audit or choose the audit beacon: every receipt carries the signed messages of its job, and anyone can recompute the ledger from the receipts. Every message is already signed by its sender in the final format, so each of those functions can be handed over without changing the client. The architecture page lists the order in which each of these is cut loose. The project's coordinator is an entrance, not a foundation.

There is no closed phase. The door opens to anyone as soon as two guards are built: files belong to jobs and can be read only by a job's parties, and limits are counted per network address, because identities are free. Everything after that is tested in the open, on the live network. There are three ways in: the program on a PC, which exists in a basic form; a browser tab with a wallet connected, for small jobs, which is planned; and gateways, which are planned.

10 Safety

The host. By design: fixed runtimes in a sandbox. No requester code. Weights in formats that cannot execute. Signed releases and reproducible builds. Limits on power, temperature and hours. Automatic pause when the owner uses the machine, and sleep when there is no work. Built so far, on Windows: only listed models run; a job runs under limits on its memory, processor, processes and lifetime; the node gives way the moment its owner uses the machine. That is not a sandbox. A job can still reach the network and the owner's files, so the only models listed today are the project's own.

The requester. Transport is encrypted, but the worker can read the request, and so can the verifier of an audited job. The network says so and does not sell privacy it cannot deliver.

Abuse. Operators choose which classes of model they serve. A gateway answers for what it lets through its door, under the law where it operates.

11 Founder independence

The project is finished when its founder is unnecessary. Eight checks define that, and each will be tested in public:

1. Existing nodes keep finding each other with every founder server off.
2. New nodes join through entry points run by the community.
3. Jobs are submitted and matched with no central API.
4. Models are fetched from other peers.
5. Payment and withdrawal run through contracts no single key controls.
6. No founder key can pause, upgrade or drain anything alone.
7. Anyone can build and ship a client from public source.
8. Protocol fixes are made and adopted without the founder.

As of this draft, none has been passed.

12 Open problems

- **Verification beyond text.** Fingerprints and tolerances for image, video and 3D models (section 6.8).
- **Adding up in size.** A model cut across home machines is limited by the links between them. It has been shown for language models. For image and video models, where far more data passes between machines at each step, it has not (section 5.1).
- **Demand.** Coins come out by the schedule whether or not anyone pays. While paid work is small, machines mine on work they give each other, as Bitcoin's miners spend power on nothing else. The first paid work still has to come from somewhere real.
- **Distribution.** An unsigned program that drives a graphics card at night looks like malware to an operating system, and signing requires a legal identity. Reproducible builds with several independent signers are the intended answer.
- **Discovery attacks.** Sybil and eclipse attacks on the hash table and on the bootstrap lists.

- **Random assignment without a coordinator.** Section 7.2 needs every peer to agree on the set of idle workers a draw is made from. Since v0.3 credit-backed jobs depend on it too.
- **A lazy or lying verifier.** One verifier's verdict destroys a bond. Known-answer jobs and dispute panels (section 6.5) are designed and not built.
- **The registry.** Who lists a model, and who removes one, once the founder cannot.
- **Cold start.** Loading a model takes tens of seconds. Routing must prefer workers that already hold it in memory.
- **Law.** A coin earned by work is treated differently in every country, and the rules change. The project sells none.

13 Changes

Since v0.3. A coordinator went live on a test chain, the first real node ran against it, and the founder made three decisions.

- There is no closed testnet. v0.3 planned one as a gate. The network is now tested in the open, and the door opens once two guards are built (section 9).
- v0.3 defined no coin and said the first phases needed none. A coin is now planned, with its cap, its schedule and its development share stated (section 7.8).
- v0.3 had a treasury take 15% of a paid job. The proposed split has none (section 7.3).
- v0.3 let paid jobs settle in an existing stable asset. Payment is now DCM only, from the payer's own wallet; a wallet address is the only identity (section 7).
- v0.3 gave units per job without a ceiling. That is safe only behind a closed door. The coin is a fixed amount a day, shared by verified work (sections 7.2 and 7.8).
- v0.3 said nothing of machines adding up in the size of a model. It is now a goal of the design (section 5.1).
- A job a worker gives back is no longer drawn for it again (section 5). Found when the first real node ran: one job went back and forth without end.

- v0.3 said the worker runs a job in a sandbox. What is built is weaker, and the paper now says exactly what it is (section 10).

Since v0.2. The message library and the coordinator were built, and building them showed where v0.2 was wrong.

- v0.2 let the client choose the worker of a credit-backed job. Most jobs are never audited, so a requester could have handed its credit to another key through a job that was never done, and credit would have been transferable in practice. Credit-backed jobs are now drawn like unbacked ones (sections 5, 7.2 and 7.7).
- v0.2 spoke of a fee on credit-backed jobs. There is none. The draw closes that loop (section 7.7).
- v0.2 defined the bond only for paid jobs, so a network without money had none. It is now held in credit, fifty jobs' worth of the job at hand (section 6.3).
- v0.2 paid verifiers only out of paid jobs. They are now paid in credit for every required audit, whatever the verdict (section 6.5).
- v0.2 had the coordinator run the audit beacon. It is now public randomness the coordinator cannot choose (section 6.2).
- Jobs expire a day after their offer (section 5). This rule is built.
- More in the architecture page: one signature rule, the body of every message, the first measured costs.

Since v0.1.

- v0.1 said that credit cannot be transferred and also that a shortfall is bought from participants with a surplus. Both cannot be true. Credit and money are now separate lanes (sections 7.2 and 7.3).
- v0.1 did not say where the first credit comes from. It comes from serving unbacked requests, which are assigned at random (section 7.2).
- v0.1 did not name the source of audit randomness and assumed an audit never misses (sections 6.2 and 6.3).
- Added the requester's audit, and what an audit reveals (sections 6.6 and 6.7).

- v0.1 implied a chain transaction per job. Jobs are too small for that; channels and batched claims replace it (section 7.8).
- v0.1 called its first phase a testnet without saying what in it is centralised. Section 9 now does.
- Added the free-entry result and the case against emissions, with a model anyone can run (sections 7.5 and 7.6).

References

1. S. Nakamoto. Bitcoin: A Peer-to-Peer Electronic Cash System. 2008. bitcoin.org/bitcoin.pdf
2. The Golem Project. Crowdfunding Whitepaper. 2016.
3. A. Baldominos, Y. Saez. Coin.AI: A Proof-of-Useful-Work Scheme for Blockchain-Based Distributed Deep Learning. 2019.
4. A. Lihu et al. A Proof of Useful Work for Artificial Intelligence on the Blockchain. 2020.
5. AI Horde. A crowdsourced distributed cluster of image and text generation workers. Totals are self-reported. aihorde.net
6. A. Borzunov et al. Petals: Collaborative Inference and Fine-tuning of Large Models. 2022. arXiv:2209.01188
7. Covenant AI et al. Covenant-72B: technical report on permissionless collaborative pre-training. 2026. arXiv:2603.08163
8. Gensyn. Verde: Verification via Refereed Delegation for Machine Learning Programs. 2025. arXiv:2502.19405
9. Prime Intellect. TOPLOC: A Locality Sensitive Hashing Scheme for Trustless Verifiable Inference. 2025. arXiv:2501.16007
10. Challenging Tribal Knowledge: Large Scale Measurement Campaign on Decentralized NAT Traversal. 2025. arXiv:2510.27500
11. drand: a distributed randomness beacon operated by the League of Entropy. drand.love