TECHNICAL WHITEPAPER

Depined v2

A compute, bandwidth and storage network run by its members

Depined turns ordinary computers into one network that can run AI models, crawl the web and store files. Models too large for any one member are cut into layer ranges and run by a team of members' GPUs. This paper describes the member node, the engine, how teams are scheduled and tested, how members talk to each other, how files are split and rebuilt, and how every layer is encrypted and verified.



646 members online

58 countries

124 GPUs

457 GB GPU memory

APPLIES TO
Desktop node 0.6.33 · Control plane 2026.11.24

PLATFORMS
Windows · macOS · Linux · Browser extension

AUDIENCE
Members, builders, security reviewers

NETWORK FIGURES
At the time of writing, 24 Nov 2026, 12:41 UTC

Contents

01	Design goals		3
02	Architecture	UPDATED	3
03	The member node	UPDATED	4
04	Workloads, the API and admission	UPDATED	6
05	Split models: one engine, many GPUs	NEW	7
06	Teams and the scheduler	NEW	9
07	A job's life and routing between members	NEW	10
08	Keeping the network healthy	NEW	11
09	Peer-to-peer mesh	UPDATED	12
10	Gossip network	UPDATED	12
11	Distributed storage		13
12	Security and privacy	UPDATED	14
13	Validated rewards	UPDATED	16
14	Releases and updates	UPDATED	17
15	The network today	NEW	17
16	Roadmap	UPDATED	17

NEW written for this revision · UPDATED changed since revision 2026.11 of 19 November 2026. All figures and thresholds describe the current release and may change.

> **WHAT CHANGED SINCE 19 NOVEMBER**
>
> The network now runs every AI model on one native engine, **shardd**, built on llama.cpp for text and stable-diffusion.cpp for image and video. The Python video engine and the whole-model text server are retired. Image and video models are split across members' GPUs in layer ranges, and a new scheduler forms teams, places spare copies and measures reliability. Jobs carry a priority and are admitted against a per-model queue cap. A network doctor tests every team every 15 minutes, a game mode hands the GPU back to members who play, and a public live view shows the network without identifying anyone.

01 Design goals

Depined has to work on the computers people already own: laptops behind home routers, gaming PCs with consumer GPUs, Macs, office machines that sleep at night. Everything below follows from six requirements.

- **Any hardware.** One engine runs on NVIDIA, AMD, Intel and Apple GPUs, and on any CPU. Each member gets the build that works on its machine, chosen by testing it there.
- **No machine has to be big.** A model larger than any member's GPU is cut into layer ranges. Each member downloads and loads only its own range.
- **No open ports.** Members only make outgoing connections. Nobody changes router settings, and nothing on a member's machine is reachable from the internet.
- **Private by default.** The directory and the public network view never list addresses or member ids. A member links only with its team partners and at most six gossip neighbours. Stored files are encrypted before they leave the network edge.
- **Verify, then pay.** Points for work are pending until the work has been checked. A member that returns wrong answers loses its pending points and its trust.
- **The member comes first.** Sharing pauses when the member plays a game, stays out of the member's home network, and never runs anything that is not signed or pinned by hash.

02 Architecture

The network has three layers. **Members** provide the resources through the desktop app and the browser extension. The **Depined Hub** is a globally distributed edge service that holds one encrypted session with every member and routes traffic to them. The **control plane** decides who does what: accounts and points, the planner and scheduler, the model catalog, admission, validation, the network doctor, storage placement and the directory. Members that work together on a model talk to each other directly over an encrypted mesh.

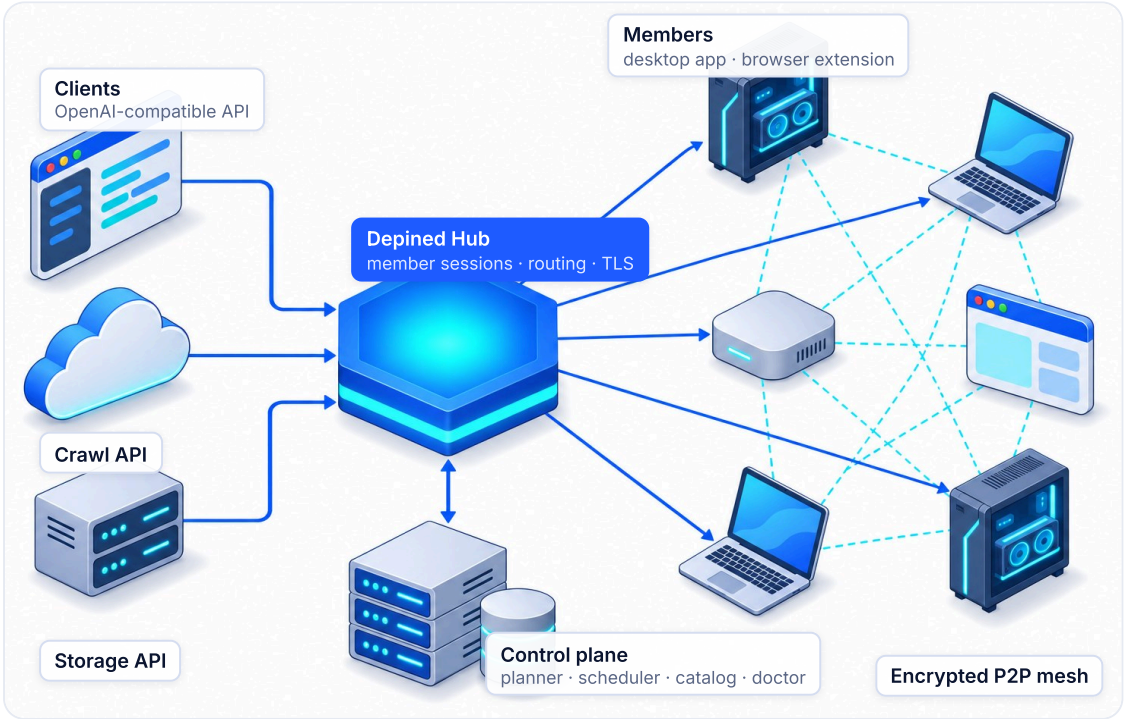


FIG. 1 System overview. Clients reach the network through the hub. The control plane plans every member's work and never accepts a request that did not come through the hub. Members only open outgoing connections: one session to the hub and encrypted links to selected peers (dashed). Diagram drawn with an AI image model; labels added by hand.

COMPONENT	ROLE	TALKS TO
Desktop node	Runs the AI engine, containers, storage and bandwidth sharing. Single Go binary, no installer dependencies.	Hub (session), control plane (plan, jobs), peers (mesh)
Browser extension	Shares bandwidth for the crawl API.	Hub
Depined Hub	Holds member sessions, routes API and crawl traffic, meters usage once a minute, relays member traffic as a last resort.	Members, control plane
Control plane	Accounts, points, planner and scheduler, catalog, admission, validation, doctor, storage coordinator, directory, gossip certificates.	Hub, members
Depined Relay	TURN relay for encrypted member-to-member links. Credentials are short-lived and issued per member.	Members
Registry, Releases, Runtime	Read-only stores: container images, signed app builds, engine bundles, and model files, including the leads' text encoders and VAEs, served from `download.depined.ai`.	Members (download only)

03 The member node

The desktop node is one self-contained program for Windows, macOS (Apple Silicon and Intel) and Linux (x86-64 and ARM64). It shows its window through the system browser engine, lives in the tray or status bar, starts at login and keeps running when the window is closed. Members choose what to share, bandwidth, CPU, GPU and storage, each with its own switch.

AI ENGINE
shardd
llama.cpp for text, stable-diffusion.cpp for image and video. Vulkan, CUDA, Metal or CPU, picked by a self-test on the machine.

CONTAINERS
Docker-compatible
Docker, Podman, OrbStack, Colima; NVIDIA GPUs inside containers.

STORAGE
Encrypted pieces
An embedded IPFS node holds erasure-coded ciphertext.

BANDWIDTH
Fetches
Public web only; the member's own network is blocked.



SCREEN 1 The desktop app: one switch to start sharing, and points for the week.

Engines from pinned bundles

Engines are not downloaded loose. The node installs **Depined Runtime bundles**: content-addressed archives whose every file is listed with its SHA-256 in a pinned manifest, one row per platform and GPU family. The app tries the GPU rows its machine supports first and always keeps the CPU build as the last resort; each install runs the bundle's own self-test before it is used, and a GPU row that fails is retried later with the reason reported. No Python and no Docker are needed for AI work.

Files are a cache NEW

Every model file a member holds, a text or image range or a lead's text encoder or VAE, is stored under its hash with a marker written only after the SHA-256 matched. The member reports all of them in its heartbeat (“held”), so the scheduler can build teams from members that need no download. Nothing is deleted on a timer. When free disk falls below the member's floor, cached files are removed by **value density**, a knapsack's greedy rule: the file whose removal frees the most space for the least loss goes first, scored as `(hours since last planned + 1) × (size in GB + 0.01)`. Files the plan still names are never removed. Downloads resume with HTTP range requests after a broken connection, retry up to six times with back-off, and still check the whole file's hash at the end.

Game mode NEW

When the member really plays a game, Depined gets out of the way: its engine processes stop so their GPU memory is freed, no CPU or GPU jobs are taken, and the control plane is told so the member's team re-forms without it. A desktop notification says sharing has paused and, two minutes after the game ends, that it has resumed.



FIG. 2 Game mode hands the GPU back. AI-drawn.

Only games pause it. Videos, browsers, editing and other GPU programs keep sharing at normal priority, because a needless pause costs the member points and the network a member. A game is detected, checked every 5 s and only after three checks in a row, as any of: a Steam game or gamescope; an exclusive full-screen Direct3D app; a program using more than 25% of the GPU whose executable lives in a game library (Steam, Epic, Riot, Xbox, GOG, EA, Ubisoft, Battle.net, Lutris, Heroic) or is a Windows game under Wine or Proton; or the full-screen app in front using the GPU heavily that is not a browser, player or creative tool.

GPU use per process comes from Windows' per-process GPU counters (the numbers Task Manager shows, any vendor), from the Linux kernel's per-client DRM statistics (AMD, Intel, open NVIDIA driver) and from `nvidia-smi`, so it works on every GPU brand. On macOS the node never pauses without evidence.

Depi, the desktop buddy

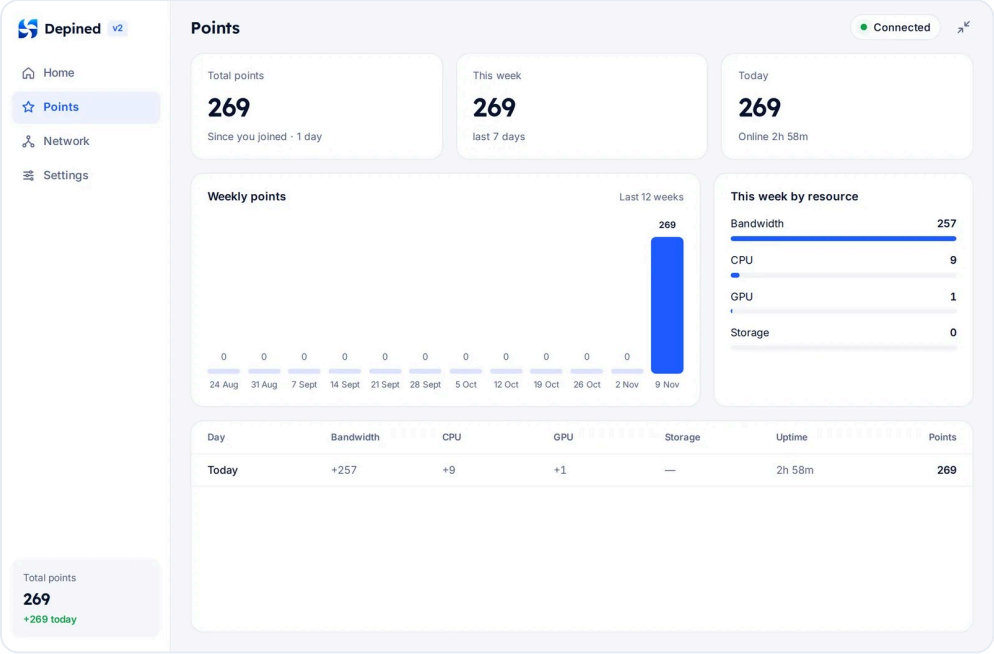
Depi is an optional animated mascot that walks along a corner of the member's screen while the app shares. Since 0.5.94 he is opt-in on the first-run screen, smaller, and stays quiet while the member is working. He cheers when a job finishes or points arrive and shows when the GPU is on an AI job. Depi runs in his own transparent overlay that only takes clicks where he actually is; the node downloads it and his animations on first use and checks each file's SHA-256.

His short lines are now written by the network's own text teams, not by a model on the member's CPU. Requests are rate-limited (one in flight per member, 12 an hour on his own, 60 an hour while the member talks to him, 8 at once network-wide) and queue behind customers' jobs. Any number he quotes, such as today's points, comes from the node's own statistics, never from the model. He knows only idle time, the time of day and the app's own events: no keystrokes, window titles or app lists.

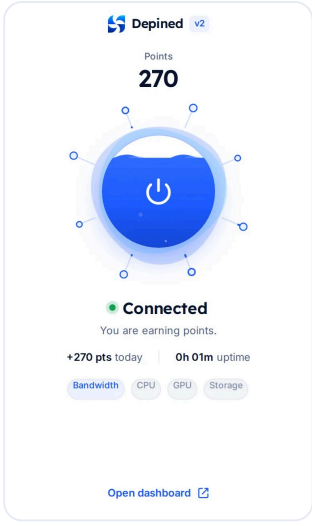
Heartbeat and plan

Every 30 seconds the node sends a heartbeat describing its hardware (GPU name, dedicated and shared memory, cores, RAM, free disk, which GPU paths passed their self-test), what it shares, which model files it holds, which ranges it has loaded and serves, its engine versions, its update setting and the result of its last

update. The answer is its **plan**: the ranges and roles it should hold, the files to fetch and their hashes, and its update channel. Apps install what their plan lists and finish running jobs before they switch.



SCREEN 2 The Points page: totals, the last 12 weeks, and this week split by resource.



SCREEN 3 The compact tray widget.

04 Workloads, the API and admission

Clients call the network with standard OpenAI-style routes: `/v1/chat/completions`, `/v1/images/generations` and `/v1/videos/generations`, using the catalog model id as the model name. A request that takes longer than about 90 seconds returns a job id to poll. Besides catalog models, the network runs **workloads**: container jobs and services with ordered variants (CUDA or CPU images, digest-pinned), of which each member gets the first its hardware can run.

Requests go only to teams that already have the model loaded. When a job arrives for a model nobody has ready, the planner is woken at once instead of waiting for its 30-second round, and the network prepares a team by itself.

Priority NEW

Every job carries a priority, and a lead always takes the highest-priority queued job first, the oldest among equals:

2 · Customers
API clients' jobs

›

1 · Own apps
the admin playground and Depi

›

0 · Checks
doctor and verifier test jobs, aimed at one member and waiting for it

Admission control NEW

A model takes only as many waiting jobs as it can work through in reasonable time. Past that, a new job is refused at once with `429 Too Many Requests` and a `Retry-After` header, instead of sitting in the queue until the caller's own timeout runs out. The network keeps finishing what it accepted and callers back off instead of piling on. Only jobs of the same or higher priority count against the cap, so checks never crowd out customers. One client may hold at most a quarter of a model's queue (at least 10 jobs), so others still get in.

JOB TYPE	WAITING JOBS PER MODEL	RETRY-AFTER
Chat	100	10 s
Embeddings	200	10 s
Image	40	30 s
Video	8	120 s
Container task	500	10 s

Text teams currently decode greedily (temperature 0), which keeps a split answer identical to the reference for each backend and makes answers comparable for validation.

05 Split models: one engine, many GPUs

Most open models no longer fit in one consumer GPU. Depined cuts a model's transformer layers into contiguous **ranges** and gives each range to a member of a **team**. Each member downloads and loads only its slice; weights never move between members, only the hidden state of the current step.

Both kinds of model run on **shardd**, a native engine that wraps an existing ggml library with a small range patch rather than re-implementing models: llama.cpp for text, so every llama.cpp model family works whole or split; and stable-diffusion.cpp (**shardd-img**) for image and video diffusion transformers.

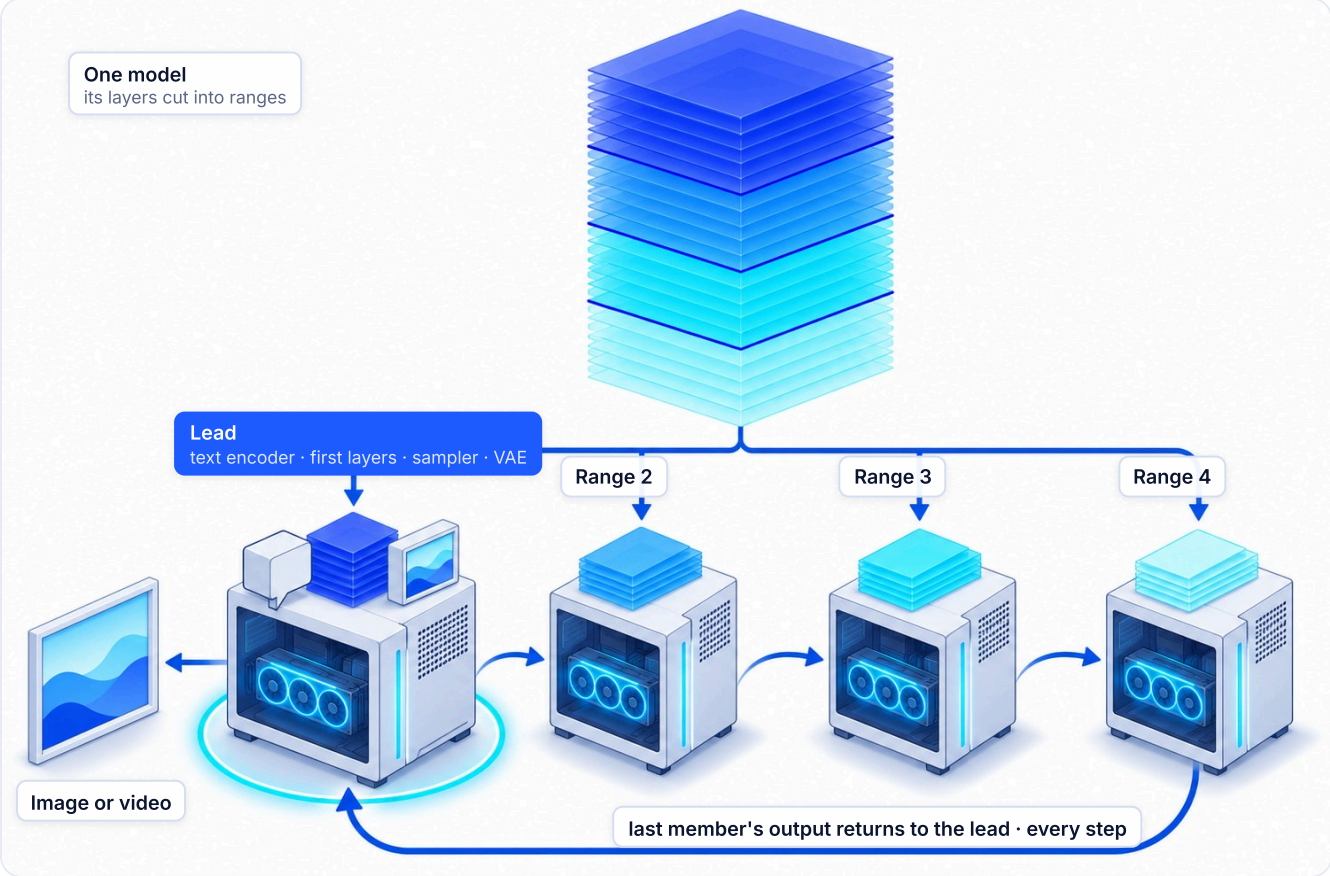


FIG. 3 A team of four runs one image model. Each member holds one slice, sized to its GPU memory. For every denoising step the hidden state passes through the members in order and the last one's output goes back to the lead, which runs the sampler and, at the end, the VAE decoder. AI-drawn; labels added.

The lead and the stages

The **lead** takes the job. It runs the text encoder, the transformer's head and its own first layers, the sampler and the VAE that turns latents into pixels or frames. Every other member is a **stage** that runs one range of layers. Stages rebuild the timestep embedding and rope positions themselves from the frame's metadata, so only one tensor crosses each hop per step. Leads need more: disk for the text encoder and VAE, and enough RAM to hold the encoder on the CPU when the GPU is short (a 16 GB RAM floor for LTX-2.5's 12B encoder). The planner checks every member's disk for its slice before giving it a place.

Models

MODEL	KIND	LAYERS	RANGES	SIZE	LEAD ALSO HOLDS	STEPS
Z-Image Turbo	Image, 6B	30	by GPU	3.7 GB	Qwen3-4B text encoder, Flux VAE	8
FLUX.1 schnell	Image, 12B	57	3	6.3 GB	CLIP-L, T5-XXL, Flux VAE	4
Qwen-Image 2.1	Image, 7B	32	3	3.9 GB	Qwen3-VL 8B encoder, own VAE	20
Wan 2.2 TI2V-5B	Video, 5B	30	3	5.1 GB	UMT5-XXL encoder, Wan 2.2 VAE	20
LTX-2.5 (distilled)	Video + audio, 22B	48	5	12.7 GB	Gemma 4 12B encoder (on CPU), video and audio VAEs	8
Qwen2.5 0.5B and others	Text	24–28	2–3	0.5–1.1 GB	tokenizer and output head (last stage)	—

Sizes are the quantized transformer (Q4 or Q8). The other text models in the catalog are SmolLM2 1.7B, Qwen3 0.6B and Gemma 3 1B. The image and video models run on members' GPUs in closed testing and are not yet offered to customers; chat is open.

What crosses a hop

An image hop carries the hidden state as 8-bit values with one scale per channel, compressed with zstd (about 18–30% smaller for about 15 ms). Rows that must not be rounded, such as a video payload's header, the text context and the timestep modulations, travel separately at full precision and are carried stage to stage exactly. For a 512×512 image on three members, a step moved about 4 MB per hidden state. Text hops carry one hidden state per token: about 130 KB for a prompt and 4 KB per generated token on Qwen2.5 0.5B.

Any GPU, checked results

The engine runs on CUDA (NVIDIA), Vulkan (NVIDIA, AMD and Intel on Windows and Linux), Metal (Apple Silicon and Intel Macs) and CPU, and a team may mix them: a range computed on an AMD card hands its output to an NVIDIA card. Every engine install runs a self-test; a member whose engine fails it leaves that engine's teams and is not planned for them again until a new engine version. The image engine computes on the dedicated GPU by name, never on a laptop's integrated one. For text, results are compared per backend: a member alone must match its backend's reference exactly, and a mixed team must agree with the CPU reference on at least 99% of positions, a near-tie within 0.5 logits counting as agreement (measured: every Vulkan and CPU miss was a tie of at most 0.28 logits).

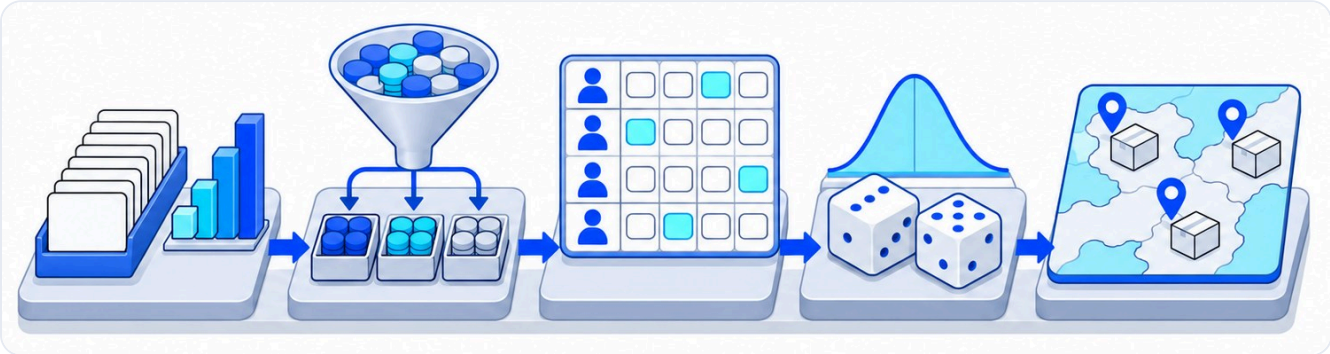
Measured on the live network

- First split chat on members (Windows, AMD RX 9070 XT + RTX 4060): correct. After limiting engine threads to half the cores, at most four, the same 7-token answer went from 38.7 s to 2.8 s.
- 50 of 50 split chats correct (AMD RX 9070 XT + RTX 3060), end-to-end p50 5.7 s, p95 7.9 s. An all-GPU Vulkan chain reached about 11 tokens per second, 48 of 48 jobs correct.

- First image from a member team: 512×512, 8 steps, 65 s, before onward passes and the later engine versions.

06 Teams and the scheduler NEW

The control plane is the only coordinator. Every 30 seconds it turns demand (jobs waiting and recent jobs) and supply (every member's heartbeat) into a plan per member. A **team** is a fixed set of members recorded in its own group record: which members hold which layers, whether all are loaded (ready), when it formed and which jobs it ran. Teams share all members or none, so one busy member can never block two teams. A job locks its whole team: a text team serves four conversations at once, an image or video takes the whole team.



1 · Demand	2 · Order	3 · Assign	4 · Reliability	5 · Spares
jobs waiting and recent use per model	models by demand; image and video before text	members to ranges and positions	sampled from each member's test record	extra copies of slices where they help most

FIG. 4 One planning round. AI-drawn; labels in the row below.

How many teams, for which models

Models are planned in order of demand, image and video teams before text teams because they need the GPU members. A model gets its base number of teams, plus one for every slot's worth of waiting jobs, up to four extra. Extra teams stay for 30 minutes after the model's last job, so a burst does not form and break teams over and over. An image or video team with no job for 30 minutes makes room when another image or video model has jobs waiting; without such demand idle teams stay ready, since nothing gains from breaking them. Established teams are planned before new ones, and every model keeps its teams before any model grows.

Assignment: a quadratic assignment problem

Position *i* in a team serves range *i*, and position 0 leads. A member's cost depends on its own position (download needed, GPU fit, quality) and on its neighbours (the measured link between adjacent positions, including the hop back to the lead). That is a quadratic assignment problem. Teams of up to **seven** members are solved exactly by trying every order (7! = 5,040). Larger teams start from the optimal assignment of the per-position costs, found with the **Hungarian algorithm** in $O(n^3)$, and improve it with **2-opt** pair swaps while the total cost goes down, as for a travelling-salesman tour. Members that may not lead are forbidden position 0. On a tie the current order is kept, so teams do not reshuffle without reason.

Reliability: Thompson sampling

Ranking a member by its average record would never give a newcomer a chance, and ranking at random would waste proven members. The scheduler treats member choice as a multi-armed bandit. Each round it draws a reliability from `Beta(passes + 1, fails + 1)` over the member's record of team tests and combines it with its network quality score: `score = 0.5 × quality + 50 × draw`. A long good record draws

high every time; a short one varies, so a promising newcomer sometimes ranks above a proven member and gets to prove itself.

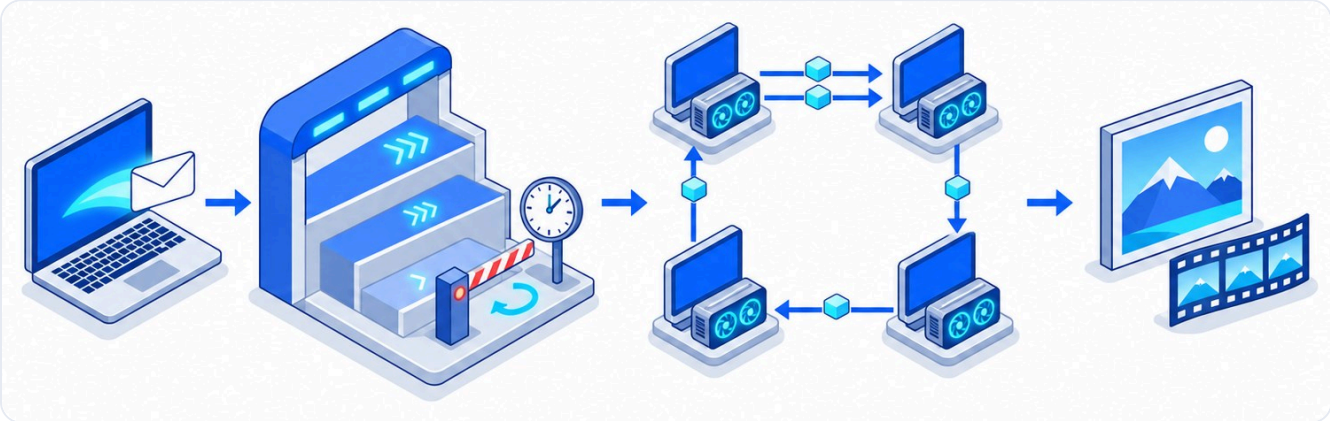
Spare copies: greedy facility location

Spare copies of slices are placed ahead of need. For each range, while it has fewer holders than its demand asks for, the member that adds the most (reliability, free room, what it already holds) receives a copy; a member takes at most a fixed number of ranges, and the loop stops when no one adds anything. Up to three image-capable members outside teams keep every slice of a model plus the lead's files on disk, so a team that loses a member re-forms in seconds with one that needs no download. Because members report what they hold, candidates are ranked by holdings first and large downloads happen only for real demand.

Members who leave

A member the hub has not seen for **45 seconds** leaves its teams; its range goes to a spare and the team re-forms, while a short network blip keeps its place. A member about to update or shut down tells its team first, and the team stops taking jobs until it has re-formed. Jobs wait for a team that is reloading instead of failing.

07 A job's life and routing between members NEW



1 · Submit
API call through the hub

2 · Priority and admission
queued by priority, or refused with 429 + Retry-After

3 · Team and hops
a ready team's lead claims it; hidden states hop member to member; slow hops are hedged

4 · Result
image or video returned, points pending

FIG. 5 The life of a job. AI-drawn; labels in the row below.

1. **Submit.** The client's request reaches the control plane through the hub. A closed or unknown model is refused at once.
2. **Priority and admission.** The job gets its priority and is admitted against its model's queue cap (section 04).
3. **Claim.** Only the lead of a *ready* team, one whose every member reports that it serves its range, may claim the job. The claim takes the team lock atomically.
4. **Run.** Each step's hidden state travels through the team (below). Every hop is timed and traced, without member identities.
5. **Result.** The lead returns the result. Points for the lead and every team member are recorded as pending until validation (section 13).

Routing: shortest paths over measured links

The member node plans a chain as a **shortest path** (Dijkstra) over layer positions, weighing each holder's measured compute time per layer, its latency and its transfer speed. One fast GPU holding many layers can

beat several slow hops, and when several members hold the same range the path picks the best one for this step.

ONWARD PASSES

Instead of returning to the lead after every member, the lead sends the frame with the rest of the route; each member forwards its output to the next itself, and the last answers the lead directly. A step takes three trips instead of four. Measured locally on 80 ms links: an image step 6.05 s → 4.59 s, a chat token 336 → 256 ms.

THE FASTER PATH PER ROUTE

Member-to-member only wins when the members' own links are good. The lead times both ways for each route (a moving average per model and member list), takes the faster and re-checks the other every 25 steps. A slow middle link falls back to hop by hop with no loss.

HEDGED HOPS

An image or video hop slower than twice its usual time (at least 3 s, at most half its deadline) is also sent to another loaded member holding the same layers. The first answer wins, the other is cancelled without counting against it, and a member that lost to the backup is avoided for the pass. Conversations are never hedged, because their stages hold the session's cache.

EXACT ROWS AND RECOVERY

A packed video payload's exact rows travel with the hidden state at every stage. A failed onward pass is redone hop by hop; a silent stage is routed around at once, or the step fails with the layer range no loaded member answers.

08 Keeping the network healthy NEW

The network doctor

Every 15 minutes the doctor tests the whole desktop network and fixes what it can, using the app's own commands pushed through the hub.

- **Text teams:** a real chat is sent to every ready team's lead, so every member holding a part is exercised end to end. A wrong answer, an error or no answer within the wait is a failure.
- **Image and video teams:** tested only when the lead can reach the whole chain, with a real image, and for video a short clip at the size real videos use (9 frames at 512×288, up to 9 minutes). A team that finished a real job in the last 30 minutes counts as passing.
- **Every other member:** app version, engine state, and roles planned but stuck.

FINDING	ACTION
Team fails once	Its members restart their stage and node, with fresh partners and state.
Team fails again	The member the error names is set aside for an hour and the team is rebuilt; the lead if none is named. Optionally, a language model reads the error and engine log and may only choose among the same fixes.
GPU fault	Device lost, out of GPU memory or a CUDA error: the member whose GPU failed is set aside, the stage named in the error or else the lead.
Lead fails 3 runs in a row	The lead is set aside, whichever partners it had.
Downloads fail certificate checks	Security software or a network intercepting HTTPS: the member is set aside instead of holding a team in “loading”.
Out of date or stuck	The app is told to update or to fetch its plan now.
Pass rate below 80%, twice	An alert goes to the operators.

Each team test also becomes a line in the member's record, the record the scheduler's Thompson sampling draws from. A team that passes clears its members' health gates.

Network quality

Every device has a 0–100 quality score that scales all its points from ×0.80 to ×1.20. For desktops it weighs uptime (30), AI work passed in doctor tests (25), connection (20), a working engine (15) and being up to date (10). Scores fall into five bands; a higher band applies at once and a lower one after 24 hours, and new devices are at least “Fair” for three days. The app shows each member its next step to improve, with a one-tap fix where possible.

09 Peer-to-peer mesh

Members talk to each other over WebRTC data channels. Every channel is encrypted with DTLS and carried over SCTP. Connection setup messages travel through the hub, bundled into one message per side. A member only accepts links from its team partners and its gossip neighbours. Data takes the best path available, direct first:

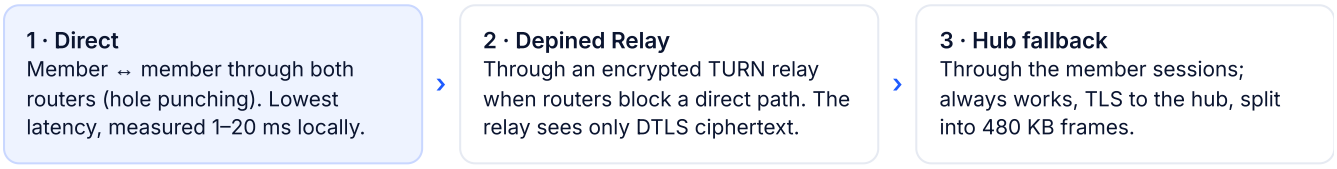


FIG. 6 The connection ladder. A link carries data only after it has answered a ping; after 45 s without an answer traffic moves back to the hub and the link reconnects later.

Each link is pinged every 20 seconds and its round trip is measured; the scheduler and the routing use these numbers. Virtual interfaces (Docker bridges, VPN and VM adapters) and link-local addresses are never offered as candidates, because they stall connection attempts. Large transfers use flow control, so a multi-gigabyte slice never sits in memory; messages are split into 60 KB chunks and duplicates are dropped when a message is resent over the hub. Hop bodies are zstd-compressed both ways when the peer supports it.

A link to a neighbour or team partner retries at most every two minutes, and the wait is forgotten as soon as the member is seen again. Before 0.6.18 the wait doubled up to 30 minutes, which left links on the hub long after both ends were back during a large rollout.

10 Gossip network

Every member should know who is online and what they hold without asking a server. Members keep links to a few neighbours and pass signed status records along them. News reaches the whole network in a few rounds, and the app draws a live graph of the member's neighbours and every team.

Choosing neighbours

Each member places the members it knows on a ring ordered by a hash of their id. Its neighbours are the members one and two places to either side, plus one a third of the way around. The offsets are symmetric, so if A is B's neighbour then B is A's. Each member has at most six neighbours whatever the network's size, and the long chords keep any two members a few hops apart. Contact lists refresh every five minutes and soon after an unknown member knocks, so members who joined later are accepted.

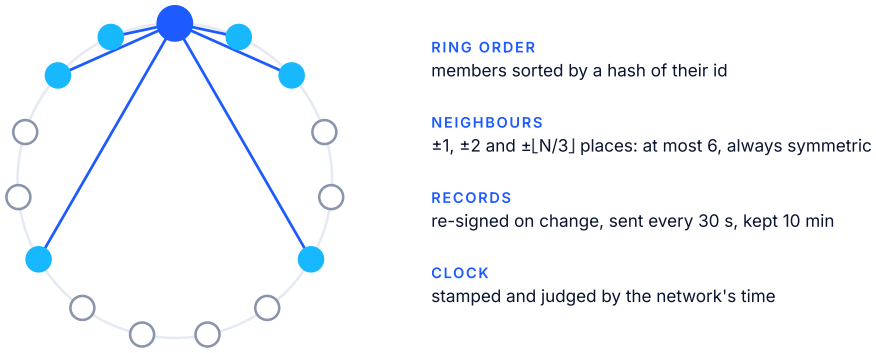


FIG. 7 A 15-member ring. The highlighted member links to six neighbours: ±1, ±2 and ±5 places.

Signed records

Each member generates its own Ed25519 key on first start. The control plane signs a certificate binding the member id to that key, valid for two hours and renewed well before it expires. A member signs a fresh record about itself (hardware, ready models, app version, links) whenever it changes and at least every four minutes, and every 30 seconds sends each neighbour only what changed. Records are stamped and judged by the network's clock, taken from the control plane's responses, so a computer whose own clock is wrong still gossips. A record is accepted only if all five checks pass:

CHECK	REJECTS
Certificate signed by the network authority	members invented by an attacker
Certificate names the record's member	a member speaking for another member
Certificate not expired	keys of members that were removed
Record signed with the certified key	records changed in transit, for example an edited GPU claim
Record at most 10 minutes old	old news replayed to make a gone member look alive

Gossip is accepted only from current neighbours or team partners, at most one message every two seconds per sender, capped at 1 MB and 400 records, and travels only over mesh links, never through the hub.

11 Distributed storage

Files stored on Depined are encrypted, cut into redundant pieces and spread across members' disks. No member holds a whole file, and what any member holds is ciphertext.

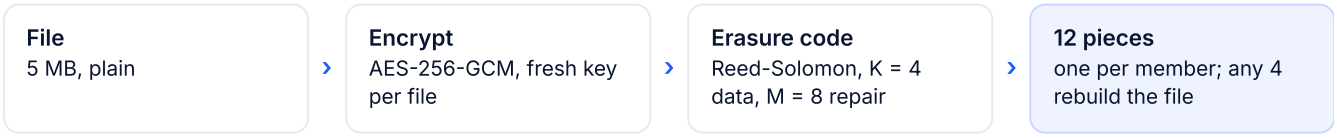


FIG. 8 A 5 MB file survives losing two thirds of its pieces.

Encrypt, then split

Every file gets a fresh random 256-bit key and is sealed with AES-256-GCM, which both hides the contents and detects any change. The file key is itself encrypted with a control-plane secret and never reaches members. The ciphertext is split with Reed-Solomon coding into K data and M repair pieces; any K rebuild it.

FILE SIZE	DATA PIECES K	REPAIR PIECES M	STORED OVERHEAD	PIECES THAT CAN BE LOST
under 1 MB	2	4	3×	4 of 6
1–10 MB	4	8	3×	8 of 12
over 10 MB	10	20	3×	20 of 30

Large files are uploaded in parts, each stored as its own segment with its own key and pieces, so a 10 GB file never has to fit in memory and downloads stream segment by segment.

Proving storage

Members are paid only for pieces they prove they hold. During a sync a member may be asked for one random block of a piece, answered as `sha256(secret nonce || block)`, which it can only compute with the block's bytes. Expected answers are prepared in advance and each is used once.

Rebuilding when members leave

The network's own storage node keeps a copy only until a file is safe: even if the member holding the most pieces disappears, members online must hold at least K + 2 different pieces. When a file falls below that margin, the network fetches pieces back (itself a retrieval check), rebuilds the missing ones from any K and hands them to new members. Pieces are content-addressed, so rebuilt pieces slot back in exactly.

12 Security and privacy

Depined runs other people's work on members' computers and moves data across the internet. Members need protection from the work, clients from dishonest members, and everyone from outsiders.



FIG. 9 The four layers of protection described in this section. AI-drawn; labels added.

Encryption in transit

LINK	PROTECTION
Client ↔ hub	TLS. API keys go in headers, never in URLs.

Member ↔ hub	TLS WebSocket. The member authenticates with a signed node pass issued at sign-in.
Member ↔ control plane	TLS with the member's bearer token. The control plane only accepts requests arriving through the hub.
Member ↔ member	DTLS on every data channel, direct and relayed.
Relay credentials	Short-lived TURN credentials issued per member. No shared static password.
Gossip records	Ed25519 signatures on every record, plus DTLS on the link.

Encryption at rest

DATA	PROTECTION
Stored files	AES-256-GCM with a fresh key per file and segment. Members hold only erasure-coded ciphertext.
File keys	Sealed with a control-plane secret. Never sent to members.
API keys and secrets	Stored encrypted (AES-GCM).
Member signing keys	Generated on the member's machine. The private key never leaves it.

Nothing runs unchecked

APP UPDATES
Ed25519-signed manifest with every file's SHA-256; signing key kept offline; macOS build signed with Developer ID.

ENGINES AND MODELS
Runtime bundles and every model slice pinned by SHA-256 and checked before use; a mismatch is deleted, never run.

CONTAINER IMAGES
Pinned by digest, so a re-pushed tag can't change what runs; allowlisted registries only.

Protecting members from the work

- **Container sandbox.** All Linux capabilities dropped, `no-new-privileges`, process, CPU and memory limits; batch jobs run with no network unless the workload needs it. Inputs are copied in and outputs out; nothing on the host is mounted.
- **Own files only.** Clean-up touches only listed names inside Depined's own data folder, and only containers labelled with the node's id.
- **Member's network is off limits.** Bandwidth requests may only reach public addresses, checked after DNS resolution, so a public name pointing at a router, NAS or cloud metadata address is refused.
- **Local app.** The app listens only on the loopback interface and requires a local-only header, so web pages can't drive it.

The public network view NEW

The landing page's live section shows the network as it is: members online, countries, GPUs and their memory, teams and finished jobs. It is built from one endpoint, cached for 15 seconds, and follows fixed rules: no member ids, IP addresses, names or exact places; members counted per country only, and a country with **fewer than three** members online left out of everything country-level (counts, teams, activity), so nobody can be singled out. Jobs show only their kind, model and time, never a prompt or an output.

THREAT	WHAT STOPS IT
Member reads stored files	Per-file AES-256-GCM, erasure-coded pieces, keys never sent to members

Member fakes results for points	Pending points, test jobs, cross-checked answers, trust score, doctor tests
Member claims storage it dropped	One-time hashed block challenges, points only for proven pieces
Malicious update, engine or image	Signed manifest, hash-pinned bundles and slices, digest-pinned images
Workload attacks the host	Capability-free sandbox, no host mounts, resource limits, no network for jobs
Client reaches a member's home network	Post-DNS private address blocking
Outsider joins a team	Mesh and relay accept only current partners; team lookups are refused to non-members
Forged or replayed status	Authority-signed certificates, Ed25519 records, 10-minute freshness
Address or identity harvesting	No IPs or ids in the directory or public view; small countries left out; links only with ring neighbours and partners

KNOWN LIMITS, STATED PLAINLY

A split model's hidden states pass through every member of its team. They are not the prompt in plain text, but they are not encrypted against the member computing them, so split inference is not confidential computing. When no mesh link is available, member traffic falls back to the hub: TLS on each leg but not end to end. Model files are public weights. The control plane holds the keys to stored files, so storage is private from members but not from the operator; client-side encryption is planned.

13 Validated rewards

Members earn points for connected time, bandwidth served, storage proven and jobs completed. Job points go into a pending ledger and are confirmed only after a waiting period during which the work can be checked. In a team, the lead is paid for the job and every team member is credited too, pending like any other work. All points are scaled by the device's network quality (×0.80 to ×1.20, section 08).

Job done
result returned

›

Pending points
held in the ledger

›

Test jobs
same input to two members

›

Agree: trust +3
points confirmed

Differ: trust −25
pending rejected

TRUST	≥ 80	≥ 50	BELOW 50	BELOW 20
Confirmation window	2 minutes	10 minutes	30 minutes	test jobs only, points withheld

FIG. 10 The reward lifecycle. Every member starts at trust 50; one wrong answer costs far more than one right answer earns.

Test jobs use fixed seeds and zero temperature. The same input goes to two members, a third breaks a tie, and the agreed answer becomes the reference for later checks. Test jobs look like any other job, so there is no way to behave well only when watched; they go only to members that are ready, and a check nobody claimed is skipped rather than failed.

OUTPUT	COMPARISON	PASSES WHEN
Chat text	Word-level longest common subsequence	similarity ≥ 0.6
Embeddings	Cosine similarity	≥ 0.97
Images	Perceptual difference hash	≤ 12 bits differ

Other jobs	Exact match	identical

14 Releases and updates

Releases are built from a tagged commit on Depined's own build machines. The pipeline compiles every platform, signs the macOS app, writes a manifest listing each file's SHA-256 and signs it with the release key. A new version goes first to a **beta ring**: members who chose “Get new versions early”. After at least 24 hours on beta, and once a ring device runs it, it is promoted: copied to `latest/`, binaries first and the manifest last, so no member ever sees a manifest whose files aren't there yet.

The control plane reads the published manifest once a minute and tells connected nodes through the hub; each checks at once, spread over 20 seconds, and also every 15 minutes on its own. A node verifies the signature and the file hash, gives a running job up to a minute to finish, tells its team it is leaving, replaces itself and restarts. The app can also be updated on demand with “Update now”. Every past version stays available, so a rollback is a matter of pointing `latest` at an earlier release.

```
tag desktop-v0.6.33 → build every platform → sign (Developer ID + Ed25519 manifest)
                    → upload releases/0.6.33/ and beta/ → verify every SHA-256 publicly
                    → ring members update → promote: copy to latest/ (binaries first, manifest last)
                    → control plane pushes "update" → members update within minutes
```

15 The network today NEW

At the time of writing (24 November 2026, 12:41 UTC) the public network view reported:

646 members online	**265 / 381** desktop apps / browser extensions	**58** countries with members online	**29** teams running
124 GPUs (55 NVIDIA, 47 Intel, 15 AMD, 5 Apple, 2 other)	**457 GB** GPU memory	**1,513** CPU cores	**19 TB** storage offered

In the 24 hours before, the network finished **2,258** chats, **53** images and **6** videos. Image and video jobs are tests while those models are in closed testing. The GPU count includes integrated graphics, which serve text; image and video ranges need a dedicated GPU of at least 4–6 GB depending on the model. Most members join from consumer laptops and desktops; the largest single GPUs in the network have 12–16 GB.

The scale changes by the hour. Current figures are on the live section of `depined.ai`, under the privacy rules of section 12.

16 Roadmap

Built, not yet switched on (image engine 8)

- **Step caching** in the image and video engine, skipping work between similar denoising steps (stable-diffusion.cpp's cache modes).
- **Several jobs in flight per team:** the lead keeps more than one copy of its own parts, so stages stay busy between one job's hops.

- **Delta-coded hops:** a step sends the 8-bit change since a recent step instead of the whole state, closed-loop so rounding never accumulates, with regular keyframes and a keyframe on request when a member is replaced.
- **Resume from a failed step** instead of restarting the job.

Next

- CPU-only members doing text encoding and VAE work for image teams.
- Opening image and video models to customers once they hold ≥ 95% success for 24 hours.
- A larger text model split across members who could not hold it alone, and audio models on the same adapter shape.
- Models as data: a manifest per model, so adding one needs no app or control-plane release.
- A replaceable lead, so a team survives losing the member that drives it.
- Client-side encryption for storage, so the operator can't read stored files either.
- End-to-end encryption for hub-fallback traffic, using the members' own keys.
- Code-signed Windows builds.

Depined v2 · Technical whitepaper DPN-WP-002 · Revision 2026.11.24. Figures and thresholds describe the current release and may change. Figures 1–5 and 9 and the cover were drawn with an AI image model and labelled by hand.