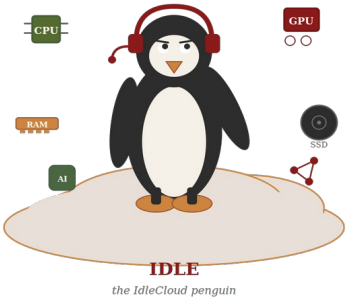
THE COMPUTE COMMONS

The Future of Personal Computing



A Declaration for Community-Owned Infrastructure

Built by and for the Linux Community

Conceived and Authored by

Ulysses Isa

Version 2.0 — May 2026

A Letter from the Author

To the Linux community,

I am not a kernel developer. I have never submitted a patch to the Linux source tree. I cannot tell you the difference between ext4 and btrfs from memory, and I had to look up how Linus Torvalds actually spelled his original kernel name before I put it in this document.

I am a lawyer. A former Senate Chief of Staff. A person who spent thirty years in the worlds of government, politics, and law — worlds where the people who build things rarely end up owning them, and the people who own things rarely built anything at all.

I came to Linux the way most people come to it: out of necessity and stubbornness. I had old hardware that the manufacturers had abandoned. Windows wanted specifications my machines could not provide. Apple wanted money I was not going to spend. So I installed Linux Mint on a 2011 MacBook Pro, then on a 2009 iMac, and discovered something I had not expected: a community that builds things because they believe those things should exist.

That philosophy is the most powerful idea I have encountered in my professional life. More powerful than any statute, any policy framework, any political coalition. A global community of people who build essential infrastructure for free, maintain it for free, and give it away for free — and that infrastructure now runs the entire world.

But I have also spent thirty years watching what happens to things that people build for free. Someone else packages them. Someone else sells them. Someone else gets the check. I have seen it happen to public policy. I have seen it happen to community organizations. And I am watching it happen to Linux right now, in slow motion, as the value migrates from the operating system to the cloud and the corporations line up to capture it.

The Compute Commons is my attempt to do something about that. Not as a technologist — I am not one, and I will not pretend to be. But as someone who knows how institutions work, how power consolidates, and how communities lose control of the things they created.

The technical architecture in this document is a starting point, not a blueprint. The community that can build a production-grade operating system kernel through mailing list arguments does not need me to tell it how to build a distributed

compute layer. What it might need is someone to say: build it now, before someone else builds it for you and charges you rent.

I do not expect to make money from The Compute Commons. I do not expect to be famous for it. I expect to hand it to a community that is smarter than me, more technically capable than me, and more stubborn than me — and watch them do what they have always done: make it work despite every reason it should not.

If I am wrong about this, the document costs nothing. If I am right, and the community picks it up, then maybe this time the people who built the infrastructure will be the people who own it.

That is all I am asking for. Build it. Own it. Keep it.

With respect and without pretension,

Ulysses Isa
May 2026

Table of Contents

You Already Won

The Linux community won the most consequential technology war in history. And nobody told them.

In 1991, Linus Torvalds was a twenty-one-year-old student in Helsinki who wanted to run Unix programs on his Intel 386. He could not afford Unix. It was proprietary. So he wrote his own kernel and released it to the world because he thought it should exist.

Four decades later, that decision runs everything. Every major cloud platform runs on Linux. Every supercomputer on the Top 500 list runs on Linux — and has since 2017, when Linux reached one hundred percent market share in that category. Android, the most widely deployed operating system in human history, runs on the Linux kernel. Every container, every Kubernetes cluster, every AI training run at every major lab — OpenAI, Anthropic, Google DeepMind, Meta, xAI — runs on Linux. The International Space Station migrated from Windows to Debian Linux in 2013. Chromecast, Amazon Fire Stick, Roku, and most smart TVs run Linux kernels. The Russian government uses Linux. The Chinese space program uses Linux.

The community that built this won completely. They beat Unix. They beat Windows Server. They beat every proprietary operating system ever made. The kernel they wrote for free powers trillions of dollars in infrastructure, handles billions of transactions per day, and runs more compute cycles than any other piece of software in the history of civilization.

And they own none of it.

Red Hat packages the kernel and sells support contracts. IBM bought Red Hat for thirty-four billion dollars. Canonical built Ubuntu and monetizes enterprise deployments. Google absorbed Linux into Android and Chrome OS. Amazon, Microsoft, and Google run it inside their cloud platforms and charge by the hour for access to infrastructure built on top of free software. Microsoft — the company whose CEO called Linux a cancer in 2001 — now makes more money from Linux than any Linux company ever has, by running it inside Azure.

The community wrote the code. The corporations collected the check.

This is not a complaint. It is a pattern. And the pattern is about to repeat — at a scale that will make the operating system wars look like a rehearsal.

The Next War Has Already Started

The operating system is about to stop mattering. Not because it failed, but because it succeeded so thoroughly that the value is moving somewhere else.

Every device is becoming a terminal. Chrome OS is already there — a browser with a login screen. Windows 365 Cloud PC runs a full Windows desktop as a virtual machine in Microsoft's data center; your laptop is just a viewport. Apple tightens the integration between device and iCloud with every release. The trajectory is clear and irreversible: the operating system moves to the cloud, the local hardware becomes a screen with a network connection, and whoever controls the cloud controls the computing experience.

When that transition completes — and it will, within the decade — the device in your hands is a screen that connects to someone else's computer. You own nothing. You compute at their pleasure, on their terms, subject to their surveillance, their pricing, and their content policies.

And your local resources — your CPU, your RAM, your GPU, your storage — will not even be serving you. They will be prey for companies that will never have enough compute. Your idle processor cycles will feed their server farms. Your storage will cache their data. Your hardware will work for them while you pay for the privilege of accessing your own desktop on their infrastructure. Your computer will be a part of their server infrastructure, and they will call it a "cloud."

The Privacy Endgame

Consider what they already do when the data is on your machine, where lawyers, regulators, and journalists can catch them. Windows 11 ships with Recall — a feature that screenshots everything you do every few seconds and indexes it with AI. Google reads Gmail to train models. Both companies collect telemetry that users cannot fully disable. That is what they do when the data lives on hardware you own, in your home, where you could theoretically find and delete it.

Now move everything to their servers. Your files. Your desktop session. Your keystrokes. Your browsing history. Your documents. Your client communications. Your medical searches. Your political activity. All of it living on infrastructure they own, in data centers you will never see, governed by terms of service they can change unilaterally, in jurisdictions they choose for legal convenience.

You cannot audit what you cannot access. You cannot encrypt what you do not control. You cannot delete what lives on someone else's disk. The surveillance is not even a policy choice at that point — it is an architectural inevitability. They do not need to snoop. They just need to not look away. The data is right there, on their

hardware or on hardware they control, and no privacy policy on earth is enforceable when the user has zero ability to verify compliance.

If the Linux community does not build a community-owned compute layer within the next seven years, it will happen anyway — but you will be compelled to pay for the privilege while having no data security and no privacy, and your hardware cannibalized by the ever-hungry compute blob. The window is closing. The time to act is here — not because the technology demands it, but because the business models do.

What The Compute Commons Is

The Compute Commons is a proposed public-benefit distributed computing platform. It coordinates underutilized storage, CPU, RAM, and GPU resources from participating devices into a shared infrastructure layer owned and governed by its community. No corporation. No shareholders. No terms of service written by lawyers whose job is to protect the company from you.

The organization is structured around a not-for-profit governance model. The Compute Commons will not generate profit for its founder. Its value is structural: the existence of a community-owned infrastructure layer changes the landscape for everyone who participates in it, develops on top of it, or is positioned to offer services adjacent to it.

The Compute Commons is not a replacement for commercial cloud infrastructure. It is the alternative that must exist so that the commercial cloud is not the only option. It is the commons before the fence goes up.

The Long-Term Vision

At its core, The Compute Commons is a resource-sharing platform. Members contribute idle storage, compute, RAM, and GPU capacity to the shared pool and draw from it when they need to. That is the product. That is what launches.

As the network matures, additional capabilities become possible — including, optionally, persistent Linux virtual machines for members who want them. A computing environment that follows your login, not your device. Your distro, your dotfiles, your tools, your data, running on community-owned infrastructure, encrypted so nobody — not even The Compute Commons — can see what you are doing. But this is an option, not a requirement. The Compute Commons is not another cloud provider selling you a desktop. It is a community sharing what it already has.

When Microsoft offers roaming desktops, it is a product. When Google offers them, it is surveillance. When the community offers them, it is sovereignty. The difference is not the technology. It is who owns the infrastructure underneath.

The Model: Free Access as a Growth Engine

Node Acquisition, Not Philanthropy

The Compute Commons provides free infrastructure access to schools, nonprofits, and Linux users of any distribution. This is not charity. It is network strategy.

Every machine that connects to The Compute Commons is both a consumer and a contributor. A school's Chromebook fleet that accesses The Compute Commons storage during classroom hours contributes idle CPU cycles overnight. A nonprofit's desktop computers that use The Compute Commons for collaboration during business hours contribute processing power on evenings and weekends. A developer's Linux workstation that pulls GPU resources for a training run during the day contributes storage and RAM while the developer sleeps.

The Sun as a Load Balancer

A single time zone sees this and worries: resources are plentiful at night and scarce during the day. But The Compute Commons is not a single time zone. It is a global network. When demand peaks in North America during daytime hours, the network routes batch workloads to nodes currently experiencing nighttime idle periods in Asia, Europe, or Africa. The sun itself becomes a load balancer. What looks like a supply problem from a single time zone becomes an advantage at planetary scale.

Not all workloads are the same. The orchestrator classifies and routes accordingly:

- **Interactive workloads** (SSH sessions, remote development, collaborative editing): latency-sensitive, pinned to the user's home region. If local capacity is exhausted, the system degrades gracefully to read-only or asynchronous operation rather than routing through high-latency intercontinental links.
- **Batch workloads** (model training, rendering, compilation, video encoding, genomic analysis): latency-tolerant, routed globally to whichever nodes are idle at that moment on any continent. These are the workloads that follow the sun.
- **Background sync** (file replication, cache warming, integrity checks): geography-agnostic, scheduled during off-peak periods across all regions.

The insight is simple: The Compute Commons does not need to solve global latency. It needs to avoid pretending latency does not exist. Workloads declare their constraints at submission time. The network routes accordingly. This is not a compromise. It is the only honest way to build a global volunteer compute layer.

Latency-Aware Replication

The Compute Commons does not know where you are. It does not want to know. What it knows is latency: when your node connects, the client pings nearby peers and measures round-trip time. The system sees that Peer A responds in 12 milliseconds and Peer C responds in 285 milliseconds. It does not know why. It does not store coordinates, IP addresses, city names, or travel history. It routes based on physics, not geography.

Your dotfiles, SSH keys, and frequently accessed tools are replicated to peers that respond within low-latency thresholds. If your latency profile shifts — because you changed networks, connected through a VPN, or moved — the system notices that different peers are now fast and begins replicating your working set to them. It is responding to network conditions, not tracking your movements. Prefetching based on observed latency, not inferred location.

The more machines on the network, the more powerful the network becomes. The more powerful the network, the more attractive it is. The more attractive it is, the more machines join. This is a flywheel, not a grant program.

Why Every Platform Is Welcome

The Compute Commons does not require Linux. It works with any operating system — Chrome OS, Windows, macOS — because The Compute Commons needs compute more than it needs ideological purity. A classroom of thirty Chromebooks represents thirty processors, thirty sets of RAM, and thirty storage drives that sit idle for sixteen hours a day. Excluding them because they run Chrome OS would be strategically irrational.

Linux users receive priority access because they are the community that will build, maintain, audit, and govern the platform. But every machine that joins makes the network stronger for everyone.

The Economics Inverted

In a traditional cloud model, the provider purchases hardware and sells compute. The Compute Commons inverts this: it gives away the service so the hardware comes to it. The “customer” is not being sold to — the customer is contributing the raw material the network runs on. No user data is harvested, sold, or monetized. The contribution is processor cycles, not personal information.

This model means The Compute Commons does not need to be profitable. It needs to be adopted.

Technical Architecture & Security

The Compute Commons is designed as a minimized-trust distributed system emphasizing client-side cryptographic control and resilient workload coordination.

Dynamic Priority Protocol (DPP)

Participation in The Compute Commons never compromises local productivity. The orchestration layer monitors local system utilization and releases resources immediately when the device owner needs them. Network tasks are migrated to other available idle nodes. The device owner always wins. Resource release is instant and unconditional. The network absorbs the loss through redundancy.

Distributed Fragmentation & Redundancy

- **Privacy by architecture:** No single participating node holds enough data to reconstruct a complete dataset or workload. Hosts cannot see the content they are partially processing. Users do not know which specific nodes handle their requests.
- **Unplug-safe redundancy:** Participants can shut down their machines at any time without affecting pool integrity. The network is designed to survive constant churn.
- **Resilient task migration:** Active workloads redistribute dynamically when participating systems disconnect or reclaim resources.

Client-Side Encryption

Encryption and decryption occur locally on user-controlled systems. Zero-knowledge architecture. User-controlled keys. No master decryption key. No central authority. No backdoor. Privacy is enforced by mathematics, not by a promise.

- XChaCha20-Poly1305 or AES-256-GCM for symmetric encryption
- Ed25519 and Curve25519 for digital signatures and key exchange
- BLAKE3 for integrity verification
- Erasure coding for redundancy and fault tolerance
- Hardware-backed authentication options (e.g., YubiKey or similar physical security keys) for high-trust node participation. The Compute Commons does not require or endorse platform-level hardware lockout mechanisms
- Cryptographic agility as a Day 1 design requirement: all asymmetric primitives (Ed25519, Curve25519) must be swappable without protocol redesign. Symmetric ciphers (XChaCha20, AES-256) survive quantum

attacks, but key exchange does not. The protocol must be architected so that migration to post-quantum schemes (CRYSTALS-Kyber, SPHINCS+) is a configuration change, not a rewrite

The Volume Thesis

Shared compute resources distributed over the internet will be slower per-operation than a dedicated data center. Certain categories of work — batch processing, distributed model training, render farms, genomic sequencing, large-scale data analysis — do not need any single node to be fast. They need many nodes working simultaneously. What any individual node lacks in speed, the network makes up in volume.

The Compute Commons does not need to compete with AWS on latency. It needs to coordinate enough idle machines to outperform what a student or nonprofit could otherwise afford — which, in many cases, is nothing.

Open Engineering Challenges

Honest acknowledgment of unsolved problems is a prerequisite for credibility:

- Scalable scheduling across heterogeneous consumer hardware with unpredictable availability
- Reputation and trust systems for node reliability in a volunteer network
- Post-quantum cryptographic migration: symmetric primitives survive, asymmetric key exchange does not. The protocol must be designed from inception with swappable cryptographic primitives
- Secure peer coordination at scale across diverse network topologies
- Achieving sufficient node density for meaningful compute capacity

These are core engineering problems. The people who built a production-grade operating system kernel through mailing list patches and IRC arguments can solve them. If they decide The Compute Commons is worth building, the engineering problems become tractable.

Public-Benefit Mission & Education

Free access is the growth engine. But the beneficiaries are real.

The K–Graduate Roadmap

- **K–12 Foundation:** Secure digital lockers for student portfolios and classroom tools, built on privacy-preserving infrastructure.
- **Undergraduate Research:** On-demand compute access for engineering, computer science, and data science coursework without prohibitive lab fees. A student at a community college gets the same compute access as a student at MIT.
- **Graduate and PhD Innovation:** GPU and RAM pools for thesis-level AI training, genomic research, and computationally intensive academic work. The dissertation should not fail because the department ran out of cloud budget.

Nonprofit Infrastructure

Most nonprofits do not need a cloud. They need reliable storage, basic collaboration tools, and the confidence that their data is not being mined. The Compute Commons offers that at zero cost, and in exchange, those organizations contribute idle compute during the hours their offices are closed.

Workforce Development

Through partnerships with initiatives like Real Tech Bros, systems retired by commercial operating system environments are refurbished with optimized Linux distributions and distributed to workforce development participants. Each refurbished machine becomes a new node on The Compute Commons network. The participant gets a functional computer with free cloud access. The network gets a new contributor. The cycle reinforces itself.

The Linux Vanguard

The global Linux community is the architect of The Compute Commons ecosystem. Not the Linux Foundation. Not Canonical. Not Red Hat. The community — the people who maintain packages nobody pays them to maintain, who file bug reports at 2 AM, who run distributions that will never have a marketing budget.

Why the Community Will Build This

Linux developers have spent decades building the software infrastructure the modern world depends on. They have watched, repeatedly, as corporations captured the economic value of their work. They built the kernel. Red Hat sold the support. They built the container runtime. Docker monetized it. They built the web server stack. Amazon charged by the hour for it.

The Compute Commons is asking them to do the same thing they have always done — build infrastructure that should exist — but this time, own it. Not as shareholders. Not as employees. As governors of a commons that cannot be acquired, enclosed, or monetized away from the people who built it.

Any Distro, Any Machine

The Compute Commons does not privilege Ubuntu, Fedora, Arch, or any other distribution. A node running Gentoo compiled from source contributes the same cycles as a node running Mint installed from a USB stick. The person running a half-working custom kernel at 4 AM is exactly the person The Compute Commons is for.

Governance, Compliance & Integrity

The Compute Commons cannot ask the community to trust it. It must be structured so that trust is unnecessary — the governance model, the code, and the cryptographic protocols must be independently verifiable.

Security Priorities

- Independent cryptographic audits as a primary funding priority
- Hardware-backed authentication through physical security keys (YubiKey or equivalent) for high-trust node participation
- Transparent governance with public decision-making processes
- Public technical review for all protocol changes and security-critical code
- Abuse mitigation and operational safeguards

Financial Model

- **Free tier (the growth engine):** Education, nonprofits, and individual Linux users access the platform at no cost. Their machines contribute idle resources. This is not a loss leader. This is the product.
- **Private sector partnerships (the sustainability mechanism):** Companies that want reliability guarantees — uptime SLAs, support contracts, compliance documentation for their auditors — are welcome to participate and their contributions help fund the free tier. However, private sector money does not purchase priority. Their workloads enter the same pool as everyone else's. The private sector is a guest in a house the community built. Welcome guests. Paying guests. But guests. They do not set the rules. They do not get a bigger chair. They help pay the electric bill.
- **Grant funding (the catalyst):** Prototype development, independent security audits, pilot deployments, and governance tooling.

Resource Contribution Valuation

The Compute Commons is exploring frameworks for transparently valuing contributed computing resources by indexing CPU cycles, GPU hours, RAM-seconds, and storage capacity against commercial market rates. Whether tax authorities will ultimately recognize contributed compute time as a deductible charitable contribution is an open question. But the possibility itself is a powerful incentive. A company with 200 desktops idle overnight does not need a guaranteed deduction to participate — it needs a credible path toward one. Contributors should consult qualified legal and tax professionals regarding any potential deductions.

Data Residency & Jurisdictional Compliance

A globally distributed network creates jurisdictional complexity. The European Union's General Data Protection Regulation requires that certain categories of personal data remain within EU borders. California's Consumer Privacy Act imposes its own constraints. Other jurisdictions will follow. If a user's erasure-coded fragments scatter across nodes in Brazil, Nigeria, and Japan, the user may be in violation of their own local data protection law without knowing it.

The Compute Commons must allow users to set data residency constraints at submission time. A European user who specifies "Keep my fragments within EU nodes only" accepts reduced performance and redundancy in exchange for legal compliance. The system honors that constraint absolutely. This is not optional — it is a prerequisite for operating legally in any jurisdiction with data protection regulation. The orchestrator classifies residency constraints alongside latency and workload type, and routes accordingly.

Content Moderation & Liability

A zero-knowledge encrypted distributed network creates an unavoidable tension: the architecture that protects user privacy also prevents the platform from inspecting what it stores or processes. If a user uploads illegal content to a system designed so that no single node — and no administrator — can see what the fragments contain, traditional content moderation is architecturally impossible.

This is not a problem The Compute Commons can engineer away. It is a legal and governance challenge that requires careful engagement with legal counsel, regulatory frameworks, and the community itself. The document acknowledges this tension honestly rather than pretending it does not exist. Hash-based filtering for known illegal content (such as CSAM databases maintained by organizations like NCMEC) can operate at the upload stage before encryption, but broader content policing is incompatible with zero-knowledge architecture by definition. The governance framework must address this openly, and the legal structure must be designed with this exposure in mind from Day 1.

Ecosystem & Future Tooling

- **Open-source productivity suites:** Decentralized document, spreadsheet, and presentation tools with privacy as the default. No vendor lock-in. No data harvesting.
- **Open-source developer tools:** Integrated development environments and cloud-native build tools accessible to student engineers without subscription fees.

- **Open-source AI tools:** Research-oriented AI compute networks for academic and independent research. The student training a model should not need a corporate sponsor to afford GPU time.

This ecosystem does not need to exist at launch. It is the long-term consequence of a healthy, growing distributed infrastructure layer. If The Compute Commons achieves critical mass, the tooling will follow — built by the same community that builds everything else.

Strategic Risks

Intellectual honesty about risks is a prerequisite for credibility and community trust:

- Achieving sufficient node participation and geographic density before the value proposition becomes self-reinforcing
- Managing distributed system reliability in a network composed of consumer hardware with unpredictable availability
- The cold-start problem: the network needs machines to be useful, but machines need the network to be useful before they join
- Complexity of secure decentralized orchestration at scale
- Regulatory and tax treatment variability across jurisdictions
- Sustaining governance discipline and mission alignment as the network scales
- Anti-anonymity legislation creating compliance requirements incompatible with privacy-preserving architecture
- Data residency laws (GDPR, CCPA) requiring jurisdictional constraints on fragment distribution, reducing performance for compliance-bound users
- Content moderation liability in a zero-knowledge architecture where the platform cannot inspect what it stores

None of these are reasons not to proceed. All of them are reasons to proceed carefully. And if the worst case materializes, the intellectual contribution remains. Ideas this directionally correct tend to get built eventually. The question is whether the community that should own it will be the community that builds it.

Strategic Roadmap

Disciplined staged growth. No promises that cannot be kept.

Phase I: Foundation

- Core architecture research and proof-of-concept orchestration
- Security model development and cryptographic protocol design
- Community contributor recruitment and governance framework drafting
- Prototype development and initial grant applications

Phase II: Validation

- Pilot deployments with controlled node participation
- University and nonprofit collaboration for real-world testing
- Governance framework formalization and community ratification
- First independent cryptographic audit
- Legal and regulatory compliance review

Phase III: Scale

- Expanded distributed compute coordination across broader networks
- Educational infrastructure partnerships at institutional scale
- Open-source tooling ecosystem development
- Research-grade distributed AI infrastructure

Phase IV: The Last Desktop

Persistent Linux virtual machines for The Compute Commons members who want them. Your computing environment follows your login, not your device. Your distro, your configuration, your data — encrypted, portable, sovereign. Accessible from any terminal on earth. Running on infrastructure the community owns.

This is the future the corporations are building for themselves. The Compute Commons builds it for everyone.

Conclusion

The Linux community built the operating system the world runs on. It built the web server the internet depends on. It built the container runtime that powers modern cloud infrastructure. Every time, someone else captured the value.

The next capture is already underway. The operating system is moving to the cloud. The local device is becoming a terminal. Whoever owns the compute infrastructure owns the computing experience. And right now, that is five companies, all running on Linux, none of them accountable to the community that made it possible.

Share compute now. Build the infrastructure now. Own it now. We will not win every battle, but we may keep whatever data security and privacy we have left.

This document is an invitation to build it first.

Conceived and Authored by

Ulysses Isa

Version 2.0 — May 2026

Technical Appendix: Conceptual Architecture

The following diagrams illustrate the conceptual architecture underpinning The Compute Commons. These are not engineering specifications — they are architectural concepts intended to communicate how the system's components relate to one another, how data flows through the network, and how the platform's growth model operates.

A.1 — System Architecture

The Compute Commons operates as a five-layer stack. Participating devices of any operating system contribute idle resources. The client handles encryption locally. The DPP ensures the device owner always retains priority. The orchestration layer handles scheduling, fragmentation, and erasure coding. At the bottom, the aggregated resource pool represents the collective capacity of all participating nodes.

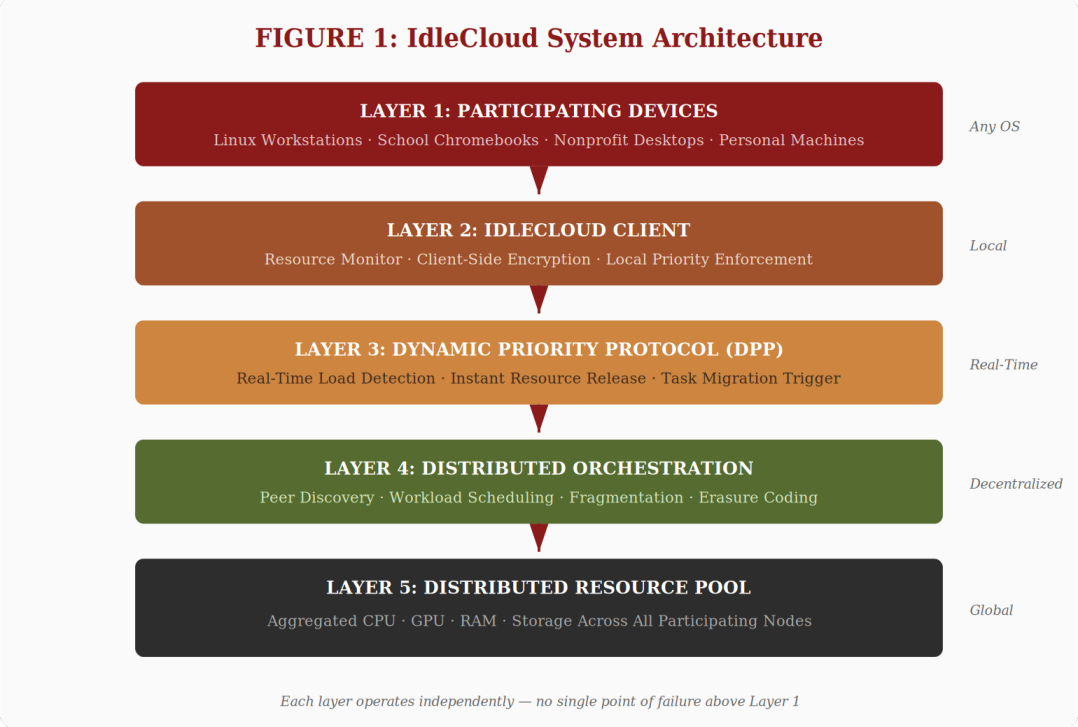


Figure 1: The Compute Commons System Architecture — Five-Layer Stack

A.2 — Dynamic Priority Protocol

The DPP cycles through six states. When a device is idle, it contributes resources. When the owner initiates local work, resources release immediately. Tasks migrate to other nodes. The device owner always wins.

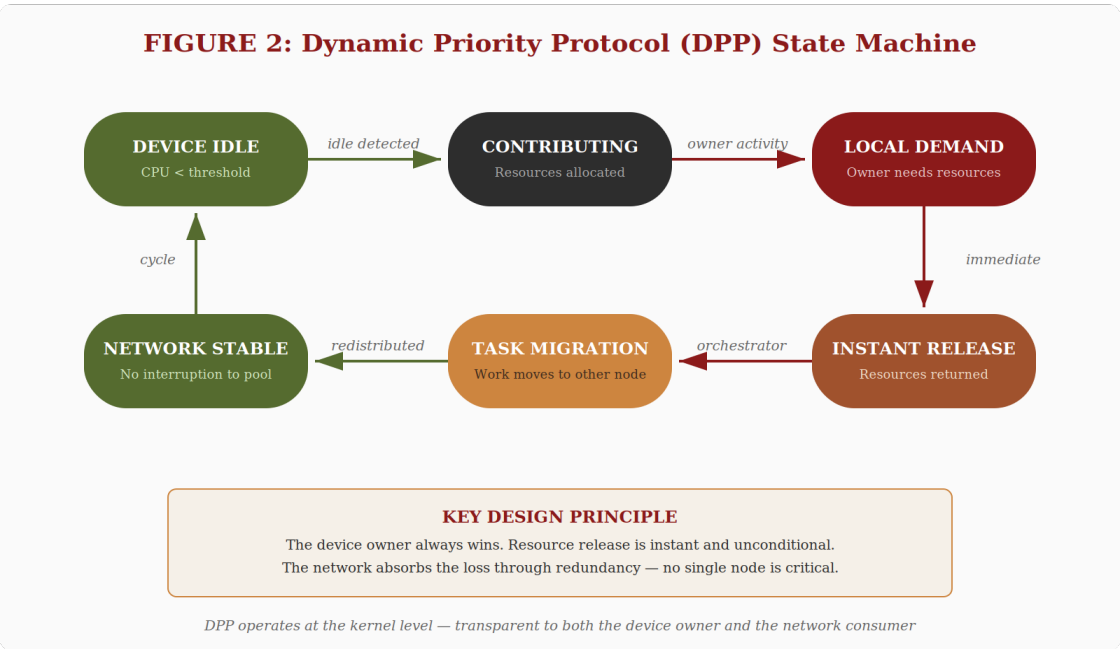


Figure 2: Dynamic Priority Protocol — State Machine

A.3 — Data Fragmentation & Encryption

User data is encrypted on the user's device before it leaves the machine. Encrypted data is fragmented and distributed across geographically dispersed nodes. Erasure coding adds parity so the original can be reconstructed from any subset of fragments. No single node holds enough data to reconstruct the original. Keys never leave the user's device.

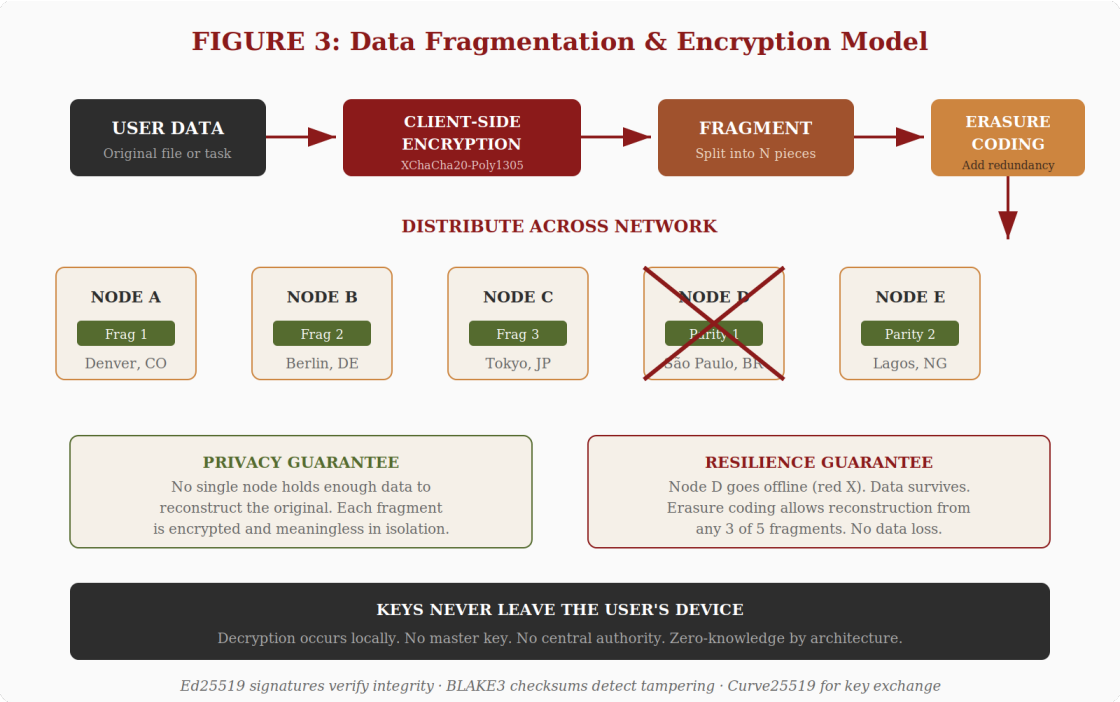


Figure 3: Data Fragmentation & Encryption Model

A.4 — The Growth Flywheel

Free access brings devices onto the network. Every device is both a consumer and a contributor. More nodes mean more capacity. More capacity makes the network more attractive. Free access is not a cost — it is the growth engine.

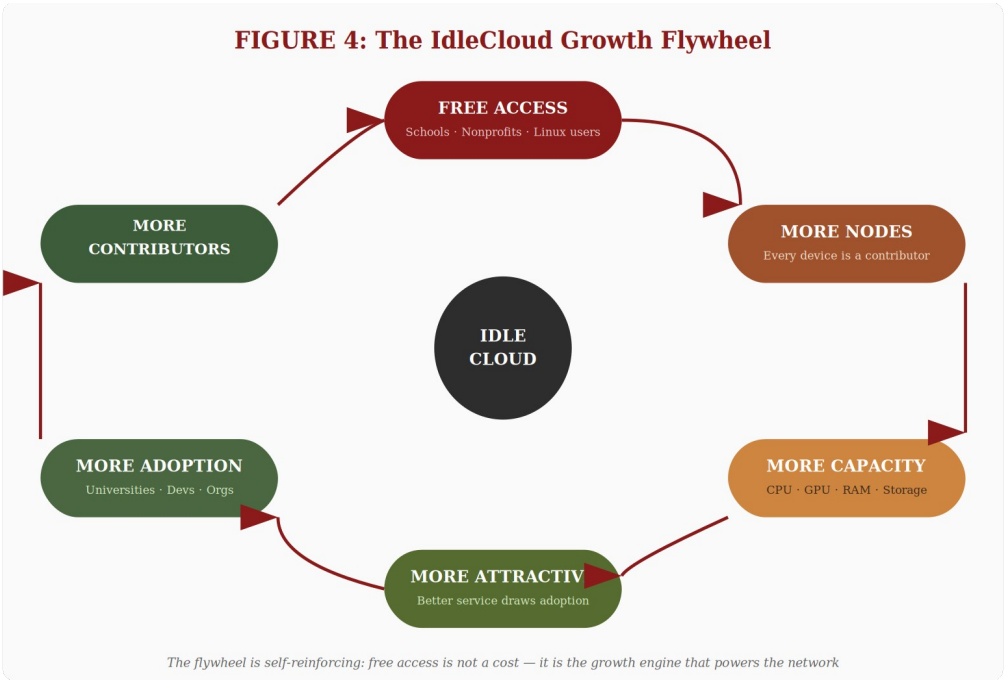


Figure 4: The Compute Commons Growth Flywheel

A.5 — Heterogeneous Mesh Network

The Compute Commons network is a mesh of heterogeneous devices: Linux workstations of every distribution, school Chromebooks, nonprofit Windows desktops, refurbished laptops, freelancers' Macs, decommissioned enterprise servers, and custom-kernel machines. The mesh routes around disconnected nodes. Resilience comes from topology, not redundant hardware.

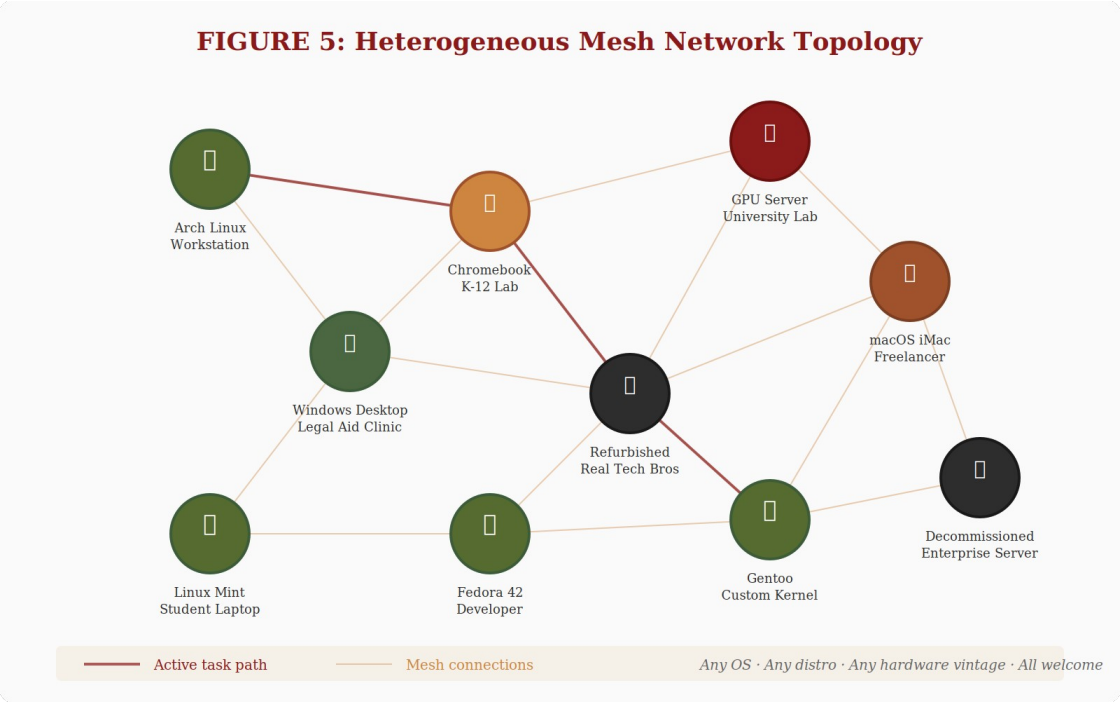
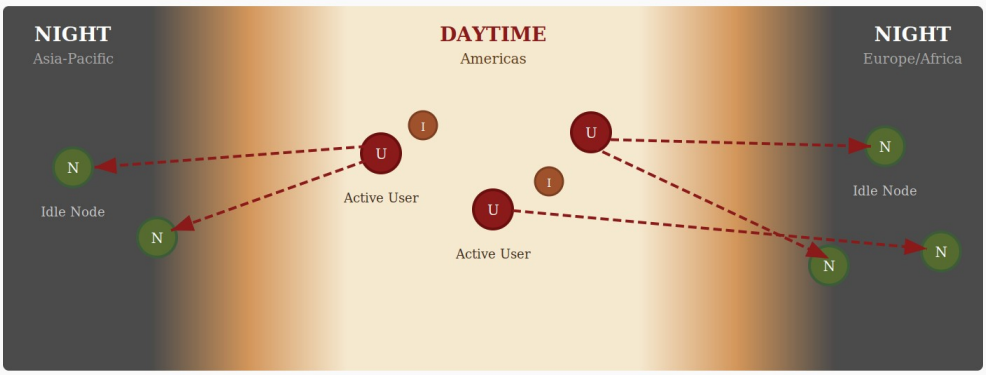


Figure 5: Heterogeneous Mesh Network Topology

A.6 — Global Workload Routing

The earth’s rotation is the scheduler. When one hemisphere sleeps, its idle nodes serve the hemisphere that is awake. Interactive workloads stay pinned to low-latency peers near the user. Batch workloads — training, rendering, compilation, encoding — route globally to whichever nodes are idle at that moment on any continent. The network does not need to solve global latency. It needs to classify workloads honestly and route accordingly.

FIGURE 6: Global Workload Routing — "It's Always Night Somewhere"



The earth's rotation is the scheduler. When one hemisphere sleeps, its idle nodes serve the hemisphere that's awake.

Figure 6: Global Workload Routing — It’s Always Night Somewhere

A.7 — Latency-Based Routing

The Compute Commons routes based on measured latency between peers, not on geographic location. The system never stores coordinates, IP addresses, city names, ISP identity, or travel history. When a node connects, it pings peers and measures round-trip time in milliseconds. Low-latency peers receive interactive workloads and replicated user data. High-latency peers receive only batch workloads that tolerate delay. The routing decision is made on physics — the speed of light through fiber — not on surveillance.

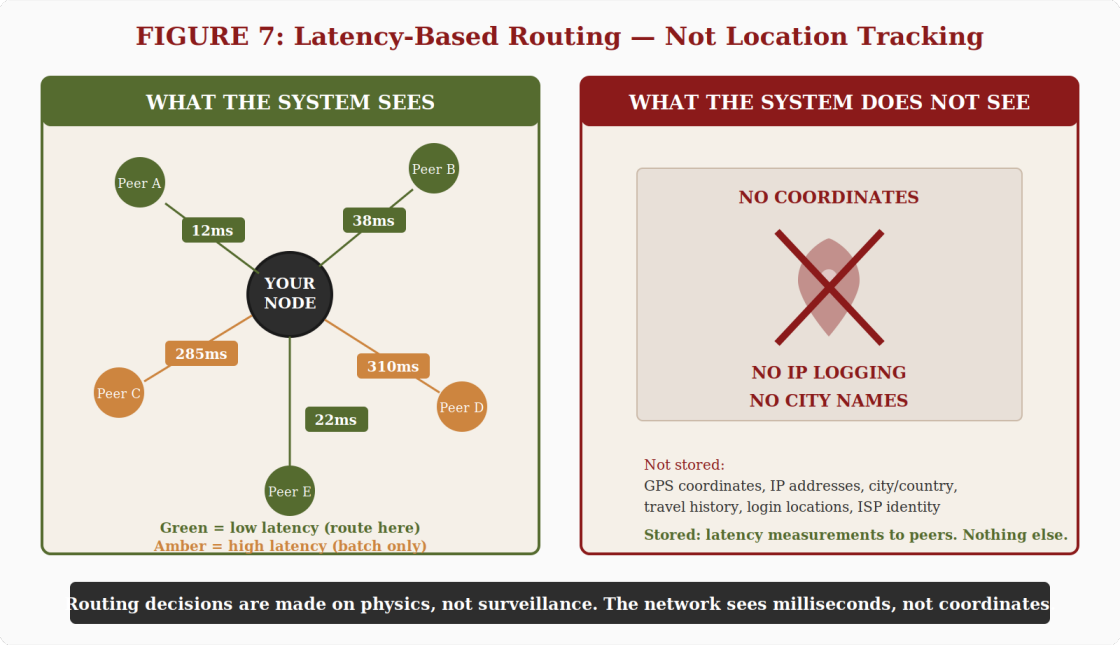


Figure 7: Latency-Based Routing — Not Location Tracking

A.8 — Technical Standards Reference

- **XChaCha20-Poly1305:** Symmetric authenticated encryption. Nonce-misuse resistant. Performant on commodity hardware without AES-NI.
- **AES-256-GCM:** Alternative symmetric cipher for hardware-accelerated environments.
- **Ed25519:** Digital signature algorithm for node authentication and message integrity.
- **Curve25519:** Elliptic curve Diffie-Hellman for key exchange between nodes.
- **BLAKE3:** Cryptographic hash function for data integrity verification. Faster than SHA-256 with equivalent security.
- **Erasure Coding:** Reed-Solomon or similar scheme enabling data reconstruction from k-of-n fragments.
- **YubiKey / Physical Security Keys:** Hardware-backed authentication for high-trust node participation. No platform lockout mechanisms.

All standards are subject to independent cryptographic audit before production deployment. Post-quantum readiness is a Day 1 design requirement, not a future migration. All asymmetric primitives must be swappable without protocol redesign. Migration paths to CRYSTALS-Kyber (key exchange) and SPHINCS+ (signatures) will be implemented as drop-in replacements when NIST standards finalize.

End of Document

Connect

Ulysses Isa

ulysses.isa@gmail.com
(201) 455-4163

GitHub github.com/Ulysses05151997
Mastodon @ulyssesisa@mastodon.social
Reddit u/Brilliant-Coffee7359
Matrix @ulyssesisa:matrix.org