HOW TO THINK LIKE A FROG

A New Theory of Networking

The FrogNet Living Network · Fawcett Innovations LLC

Why This Matters to You

The rest of this document is for network engineers. This section is for everyone else. It is shorter than the rest on purpose — you do not need the technical detail to understand what FrogNet does or why you might want one.

A Small Box That Changes What Your Family Can Do

Imagine a small box — about the size of a deck of cards — plugged in at your house. Your mother has one at her house. Your sister has one at her flat in London. Each box costs about what you'd spend on a nice dinner out.

Plug them in and they find each other. You do not configure anything. You do not set up accounts. You do not pick a service plan. The boxes form a private, encrypted network that runs a photo gallery, a family chat, a shared calendar, a bulletin board for sticky notes, a family map that shows where everyone is, and video calling — all on hardware that belongs to your family and nobody else.

Your photos are not on iCloud. Your messages are not on WhatsApp. Your location is not sold to data brokers. Everything stays on your boxes. The only thing that crosses the internet is encrypted traffic that nobody — not your internet provider, not a hacker, not a government — can read, and that does not look like FrogNet to anyone watching.

The ongoing cost is a few dollars a month[1] — a small broker subscription that lets your phones reach the network when you're away from home. That is all. No per-person fees. No storage tiers. No annual price increases. No surprise changes to the terms of service. A small broker subscription, and your family has its own private internet — internationally — for as long as the boxes are plugged in.

[1] The actual cost depends on how the broker is deployed. A typical family network running on a commercial broker service is expected to run a few dollars a month — primarily data transfer for the traffic that crosses the internet between your boxes. A family running the broker on their own hardware (for example, a small server already at home or in an office) pays essentially nothing beyond electricity. Larger or more active networks cost more; the current FrogNet

development network, with ten nodes across Seattle, New York, and Amsterdam, running heavy testing traffic, runs about forty dollars a month. A typical Mom-and-family deployment will be on the low end of that range, not the high.

What You Pay Today for the Same Things

Most people are already paying for all of this. They are just paying a different company for each piece, and their data is spread across those companies. Here is what a typical family pays today for the services a single FrogNet box would replace, and what each service costs on FrogNet.

What You Do	What You Use Today	Typical Cost	How FrogNet Does It	FrogNet Cost
Store family photos	iCloud+, Google One, Amazon Photos	$3–$10 / mo	Private gallery on your box	Included
Know where family is	Life360 (Gold), Find My	$15 / mo	GPS map in the FrogNet app	Included
Monitor your home	Ring, Nest, SimpliSafe	$10–$20 / mo	Sensors on your box, no cloud	Included
Check on an aging parent	Life Alert, medical alert services	$30–$60 / mo	Passive sensors + family alerts	Included
Share files with family	Dropbox, Google Drive, OneDrive	$10 / mo	Peer-to-peer, on your hardware	Included
Message your family	WhatsApp, iMessage, Signal	Free but with tradeoffs	FrogChat on your box	Included
Video call family	FaceTime, Zoom, Meet	Free with limits	Peer-to-peer, no call server	Included
Shared calendar	Google Calendar, Apple Calendar	Free, but your schedule trains AI	Calendar on your box	Included
Typical family total, today	**$60–$120 / mo** *+ unlimited access to your personal data*		**Total on FrogNet**	**A few $ / mo** *+ your data stays yours*

Look at the bottom row. The right side is the whole bill on FrogNet — one broker subscription, for every person in your family, everywhere you live, for as long as the boxes stay plugged in. Your data stays on hardware you own.

The left side is what you already pay — money leaving your household every month, plus a second price that never shows up on the bill: unlimited access to your personal data. The companies in that column read your messages, scan your photos, sell your location, and train their AI on your schedule. You pay them, and then they monetize you on top. FrogNet is the offer to stop paying both prices.

Mom Living Alone

If there is one scenario that explains why this matters, it is this one.

An elderly parent lives alone. You want to know she is okay. The options on the market are not great — cameras she will hate, wearables she will forget, services that charge sixty dollars a month to watch her.

A FrogNet box at her house, with a few small passive sensors, learns her daily patterns. When Mom gets up. When she makes coffee. When she moves through the house. It does not watch her. It does not record her. It notices when the pattern changes. If Mom does not get up at her usual time, the system alerts the family — through the private FrogNet, not through a company's servers. No cameras. No surveillance. Her dignity stays intact. Your peace of mind goes up.

That one use alone is worth the cost of the box.

It Still Works When the Internet Does Not

Every app on your phone assumes the internet is there. When it is not — a storm, a blackout, a rural road with no bars — your apps are useless. You cannot call, text, or check on anyone.

FrogNet does not assume the internet is there. Each box is a complete, self-contained network. If the internet disappears, the box keeps running. If two boxes can see each other by WiFi or Ethernet, they mesh together and share everything — photos, messages, sensor data, GPS positions — with no internet required. When the internet comes back, everything catches up.

This matters most in the moments you cannot plan for: when a hurricane takes out the cell towers, when your kid is at a camp with no coverage, when an ambulance drives through a dead zone with a patient in the back. The same box that handles your family photos in good weather keeps working in bad weather. That is not a feature anyone else offers.

For a longer look at how FrogNet performs in emergencies, see Appendix A at the end of this document.

A Word About Privacy

People ask how FrogNet compares to Signal, or to a VPN. The short answer is that Signal is excellent at what it does, and a VPN is useful for what it does, but neither one does what FrogNet does.

Signal protects the content of your messages from the people carrying them. FrogNet eliminates the carrier. There is nobody in the middle to trust, compromise, or compel. No servers to subpoena. No phone numbers. No accounts. No metadata. And it keeps working when the internet does not.

A VPN is an encrypted tunnel to someone else's building — the cloud, Google, Apple, Meta. Your data still ends up on their servers. A FrogNet is not a tunnel. It is your own building. The services run on your hardware. There is no "other end" where a company has your data. There is no

company at all.

What Follows

Everything described above is operational today. FrogNet runs across ten nodes in Seattle, New York, and Amsterdam, with more locations coming online. On April 17, 2026, a collaborator in New York completed the first independent end-to-end closed-loop demonstration — a web application, a sensor, and a physical actuator, all on a WiFi network in New York, coordinating through a database 2,400 miles away in Seattle, entirely over FrogNet's semantic protocol. It is not a prototype. It is not a simulation. It is a production system built over more than a decade by a network architect with fifty years of experience at companies including Boeing, Electronic Arts, Sierra On-Line, Microsoft, and NanoString Technologies.

The rest of this document explains how it works. It describes twelve paradigm shifts — foundational assumptions about how networks have worked since 1969 that FrogNet replaces — followed by a comprehensive inventory of everything FrogNet is and does. If you are not a network engineer, you do not need to read it. If you want to understand why the claims in this section are architectural truths rather than marketing promises, read on.

Introduction

Every network in existence today was built on a set of assumptions that made sense in 1969. Connectivity is rare and expensive, so design for disconnection tolerance. Memory is scarce, so keep the network stateless. Administrators are available, so require human configuration. The internet is reachable, so treat it as the backbone.

Those assumptions have calcified into orthodoxy. Fifty years of protocol design, tooling, and engineering education have been built on top of them. Nobody questions them because everybody learned from people who learned from people who learned from people who never questioned them either.

FrogNet questions all of them.

This document describes twelve paradigm shifts — foundational assumptions about how networks work that FrogNet replaces with better answers for the environments it actually operates in. It then describes, in detail, everything FrogNet is and does.

Reading this document will not make you a FrogNet engineer. But it will make you think differently about what a network can be — and that is the prerequisite for understanding why FrogNet matters.

The Paradigm Shifts

FrogNet did not make incremental improvements to existing networking. It went back to first principles and replaced twelve foundational assumptions with designs that match the actual constraints of the environments it operates in.

1. Transport Abstraction — Protocol Independent of Physical Layer

Every network stack in existence is built around a specific transport assumption. TCP/IP assumes reliable-ish packet delivery. Radio protocols assume unreliable, low-bandwidth, high-latency links. WiFi assumes high-bandwidth local area. The entire OSI model is built around the idea that you choose your transport and build your protocol for it.

FrogNet separates transport from processing completely. The application layer has no idea what transport is underneath. The same code runs identically over fiber, WiFi, Ethernet, WireGuard tunnel, LoRa, satellite, or a narrowband RF link. You don't configure for the transport. You don't optimize for it. You don't even know it's there. The semantic compression layer adapts automatically to whatever the link gives it.

This isn't just transport agnostic as a marketing claim. It's an architectural decision that eliminates an entire category of protocol design work. It also means that a single deployment can span WiFi, Ethernet, and RF simultaneously — the same network, the same code, the same behavior, regardless of what the physical layer is doing underneath.

One line worth stating explicitly: the radio is just a pipe. Which radio? Whichever one is there. FrogNet does not care, and FrogNet is not a radio protocol. The conclusions in this document about performance, compression, and survivability are consequences of what happens *above* the transport, not of any particular choice of transport below it.

2. Offline-First — Disconnection Is the Baseline, Not the Error

Every networked application ever built assumes connectivity as the baseline and treats disconnection as an error condition to be handled. Offline mode is a feature you add later, grudgingly, when users complain. Progressive Web Apps, service workers, local storage — these are all retrofits on top of a fundamentally online-first architecture.

FrogNet inverts this. Offline is the default. Every node operates fully standalone as the baseline condition. Connectivity is an enhancement that arrives when available and disappears without consequence. Applications built on FrogNet don't handle disconnection — they never assumed connection in the first place. The architecture matches the physical reality of the environments FrogNet operates in, where connectivity is intermittent by nature.

3. /etc/hosts vs. DNS — Ground Truth Over Administered Namespace

DNS was designed for a world where networks are large, stable, and administered by professionals. You configure a nameserver, you delegate zones, you manage TTLs, you wait for

propagation. The assumption underneath all of it is that the network topology is known, relatively static, and has a central authority responsible for keeping the namespace consistent.

FrogNet operates in the opposite environment. Nodes appear and disappear. Topology changes without warning. There is no central authority. There is no administrator. The network manages itself or it doesn't work.

DNS fails in this environment — not because it's poorly implemented, but because it was designed for a different problem. A nameserver that goes offline takes its zone with it. A TTL that hasn't expired serves stale data. Split-brain scenarios during network partitions produce inconsistent resolution across the mesh.

FrogNet uses /etc/hosts — the oldest, simplest, most reliable name resolution mechanism in existence. Every node maintains its own hosts file. runMerge recalculates and rewrites it every time the topology changes. There is no TTL to expire, no zone to delegate, no nameserver to go offline. The resolution is local, instant, and always current for the topology the node can actually see.

The floating databasehost.frognet address — always the highest-IP active node on the mesh — is recalculated and rewritten to every hosts file on every merge. No node ever has a stale address for the database. The moment the topology changes, the hosts file changes with it.

DNS is the right answer for a stable, administered network. /etc/hosts is the right answer for a self-forming mesh where the only ground truth is what the node can directly observe. FrogNet uses the right tool for the actual problem.

4. Sovereign Networking — No Dependency Chain, No External Authority

Every network you use today depends on someone else's infrastructure. Your WiFi depends on your ISP. Your ISP depends on backbone providers. Your applications depend on cloud providers. Your DNS depends on root servers. At every layer, someone else is in the chain — someone who can fail, someone who can be compromised, someone who can cut you off, someone who can surveil you, someone who can monetize your traffic.

FrogNet has no such dependencies. The network is yours. The data is yours. The infrastructure is yours. No ISP, no cloud provider, no DNS authority, no certificate authority, no backbone provider is in the chain. You are not a tenant in someone else's infrastructure — you are the infrastructure.

This matters differently in different contexts. For a family it means privacy. For a business it means control. For a military unit it means survivability. For a community it means resilience. But the underlying shift is the same: from dependent networking, where you operate at the sufferance of others, to sovereign networking, where you operate on your own terms.

5. Emergent Internet — Self-Assembled, Not Designed

The internet was designed. IANA allocates address space. ICANN manages domains. ISPs negotiate peering agreements. BGP routes are configured by human administrators. The entire global routing infrastructure is the product of deliberate human decisions made by known authorities.

FrogNet's internet emerges. No allocation authority. No routing administrator. No peering negotiation. Nodes discover each other, calculate topology, write routes, and form a coherent network — automatically, repeatedly, every time the topology changes. The internet that results wasn't designed by anyone. It assembled itself from the nodes that happened to be reachable at that moment.

This is a fundamentally different model of what a network is. The traditional internet is a constructed artifact. A FrogNet is a living system.

6. Stateless vs. Stateful — The Network Carries Context

Every network built since the early days of the internet is stateless by design. Each request arrives with no memory of what came before. The server reconstructs context from scratch — parsing headers, re-establishing session state, re-transmitting data the other end almost certainly already has. This was a reasonable choice in 1969 when memory was scarce and connections were unreliable. It has been the default assumption ever since, and nobody questioned it.

FrogNet questions it.

The semantic compression engine introduces state at the network level. The fabric remembers what it has already seen. Templates are learned from the structure of real traffic and cached locally. When a response arrives that matches a known template, only the values that changed cross the wire — not the structure, not the unchanged fields, not the overhead. The network itself

carries the context so the application doesn't have to re-transmit it.

This is not delta encoding. Delta encoding sends the difference between two versions of a document. BLDC-1 goes further — it separates structure from value, learns the structure once, and then only transmits values. On steady-state traffic, a 500KB JSON payload becomes 50–150 bytes on the wire. The application never changes. The protocol never changes. The wire changes completely.

The stateless model was a constraint imposed by 1960s hardware. FrogNet removes the constraint.

7. Infrastructure Compression — Transparent, Automatic, Zero Application Changes

Every compression system ever built is an application-level decision. You choose to compress a file. You choose to gzip an HTTP response. You configure your CDN to minify assets. Compression is something developers do deliberately to specific data when they decide it's worth the effort.

FrogNet makes compression an infrastructure property. The developer writes REST. The network compresses everything, automatically, transparently, using a codec that learns the structure of the actual traffic flowing through it. There is no configuration, no decision, no per-endpoint optimization. Every HTTP exchange on every interface is compressed. The developer never thinks about it.

This is the same paradigm shift that happened when hardware TCP offloading moved network stack processing from software to silicon — except FrogNet does it for semantic content, not just packets. All system-level coordination within FrogNet uses REST, which means the coordination overhead itself is compressed. On a steady-state network, management traffic approaches zero wire cost.

8. Serial vs. Parallel — Physics as the Only Constraint

Traditional request/response networking is serial at its core. A request goes out. The sender waits. The response comes back. The next request goes out. Even with pipelining — which overlaps the sending of requests — processing remains sequential and the wire sits idle between transactions.

On a constrained link, idle time is catastrophic. Every microsecond of dead air is bandwidth you cannot recover. A 4800-baud link is already at the edge of what's usable. Wasting half of it on dead air between serial transactions makes it unusable.

FrogNet eliminates the dead air entirely.

The 256-worker daemon pool means up to 256 requests are in flight simultaneously — not sequentially overlapped, but genuinely parallel, each on its own thread, each with its own database connection. By the time the wire finishes transmitting one frame, the next is already queued and waiting. The transmitter never starves. The link runs at or near 100% duty cycle continuously.

The result is that effective throughput is not limited by processing speed — it is limited only by physics. You cannot move bits faster than the channel allows. FrogNet makes sure every bit the channel allows is carrying useful payload.

This is why the measured throughput improvement is 44× on the same physical link. The link didn't get faster. The utilization went from fractional to near-total. The measurement was taken on an Ethernet-to-Ethernet channel shaped with delay, jitter, and loss to match a narrowband RF condition — a real 4800-baud throughput ceiling, achieved deterministically.

9. Networked Intelligence — Distributed AI with Shared Real-Time World State

Every AI deployment ever built follows the same model. Data is collected, sent to a central location, processed by a model, and results are returned. The AI lives in the cloud. The sensors live at the edge. The network is the pipe between them. The AI is smart but blind — it can only see what gets sent to it, with whatever latency the pipeline introduces.

This model has three fundamental problems. It requires connectivity. It introduces latency between observation and action. And it concentrates intelligence in one place, which means a single point of failure, a single point of compromise, and a single point of control.

FrogNet dissolves all three.

Every node can run its own AI. Every AI has direct, real-time, read-write access to the transient database — which means every AI can see every sensor on the entire mesh, not just the sensors on its own node. Multiple AIs can run simultaneously on different nodes, observing the same data, writing to the same scratchpad, coordinating through the transient database without any message-passing infrastructure, without any API contracts between them, without any orchestration layer. They share a world. They act in it.

Multiple dashboards can do the same thing — any node, any browser, any connected device sees the same real-time network state simultaneously. There is no master dashboard and no replica. There is one transient database and every observer sees it directly.

The sensor-to-actuator loop — sensor writes to transient DB, AI reads it, AI writes control signal, actuator reads control signal — runs entirely within the mesh. It survives complete internet loss. It survives the loss of any individual node. It survives network partitions, because each fragment continues operating on the sensors it can reach. The first independent end-to-end demonstration of that loop was completed on April 17, 2026 — a button press in a Flask UI on a WiFi network in New York, routing through the transient database in Seattle, activating a physical actuator back in New York.

The traditional model puts intelligence in the cloud and data at the edge, connected by a fragile pipe. FrogNet puts intelligence at the edge with the data, connected by a mesh that heals itself. That is not an improvement on centralized AI. It is a different theory of where intelligence lives in a network.

10. Network as Database — Data in the Network, Not Behind It

Every distributed system ever built treats the network as a pipe — data lives in databases, the network moves it between them. You have servers with databases, clients that query them, and a network that connects them. The database and the network are separate concerns.

FrogNet collapses that distinction. The transient database isn't behind the network — it is the network. Every node can read and write it. Every sensor writes to it. Every dashboard reads from it. Every AI host queries it and writes back to it. There is no client-server distinction because there is no separate tier for data storage — the data is in the network itself, always available, always current, always at the same address regardless of which physical node is hosting it at any given moment.

This is what Gelernter was reaching for with Linda tuple spaces in 1985. FrogNet is the first practical distributed implementation of that idea at network scale.

11. Living Topology — Derived from Reality, Never from a Plan

Every network you administer has a topology you designed, documented, and maintain. Subnets are planned. Routes are configured. VLANs are assigned. When something changes you update the documentation, push new configs, and hope nothing breaks. The network is a designed artifact that requires human maintenance to stay consistent with reality.

FrogNet's topology is never designed and never documented because it doesn't need to be. It is calculated fresh from observed reality every time anything changes. The highest-IP node becomes the database host. Routes are written to match whatever interfaces are actually up. Hosts files reflect what the node can actually reach. The network's representation of itself is always derived from ground truth, never from a plan that may have drifted from reality.

Network engineers spend enormous effort keeping designed topology consistent with actual topology. FrogNet eliminates that gap by never having a designed topology in the first place.

12. Security — Physical Proximity Replaces Perimeter Defense

Every traditional security model assumes a hostile perimeter. The internet is outside, attackers are out there, and you build walls to keep them away from your services. Every service behind those walls still needs its own authentication layer because the perimeter is assumed to be already breached.

FrogNet rejects that premise entirely.

The physical layer is the first perimeter. You cannot attack a FrogNet from across the internet — it isn't on the internet. You must be physically present and in range to reach the mesh. This isn't a policy decision — it's physics. No firewall rule, no certificate, no password protects you the way physics does.

Network membership is the trust boundary. If you're on the mesh, you're trusted. If you're not, you can't reach anything to attack. Layering traditional per-service authentication on top of this is not

just unnecessary — it's the wrong mental model applied to the wrong architecture.

This means the complexity, latency, and overhead of PKI, TLS handshakes, session tokens, and credential stores simply don't exist at the application layer. That overhead — which can consume 25 seconds of a 4800-baud link just for a TLS certificate chain — is eliminated by design, not by cutting corners.

WireGuard handles tunnel encryption where internet traversal is required — Curve25519, compact handshake, solid crypto. The security model is layered correctly: physics first, WireGuard where needed, standard Linux security tooling available underneath for anyone who wants it.

Traditional security paradigms were built for systems exposed to the internet. FrogNet is not that system.

What FrogNet Is and Does

The following is a comprehensive description of FrogNet's architecture, capabilities, and proven performance characteristics as of April 2026.

Network Foundation

- Transport-agnostic fabric — WiFi, Ethernet, WireGuard, LoRa, narrowband RF, any combination
- Self-forming — nodes discover each other automatically, no configuration required
- Self-organizing — topology managed without human intervention
- Self-optimizing — routing and compression improve as the network learns traffic patterns
- Self-healing — network fragments operate independently, rejoin without reconciliation
- Network-to-network architecture — not device-to-device, entire networks federate
- No central authority — no single point of failure anywhere in the design
- Internet-optional — uses internet when available, operates fully without it
- Federation — even extremely low-speed links cause independent networks to join and behave as one cohesive system

Semantic Compression Engine (BLDC-1)

- Reduces structured HTTP traffic by 93.8% in production
- Supports HTML, XML, JSON, CSV, and plain text natively
- Extensible via OO derivation — new data types added without changing the core
- Template learning — network learns structure of traffic automatically
- SAME/DIFF protocol — unchanged responses collapse to 16 bytes
- Applied at infrastructure layer — zero application changes required
- REST is the native protocol — coordination and data use the same compressed transport
- 44× throughput improvement proven on constrained links (Ethernet-shaped to match narrowband RF: 4800 baud, added jitter, 20% packet loss)
- Parallelization fills dead air between messages — near-100% link duty cycle
- All system-level coordination uses REST — automatically benefits from compression

Node Architecture

- Each node is a complete sovereign network
- Runs DNS, DHCP, Apache web server, MySQL database locally

- Operates fully standalone with zero connectivity
- Meshes automatically when other nodes come into range
- Standard Linux hardware — Raspberry Pi, x86, anything
- Python, PHP, bash — no exotic dependencies

Transient Database

- Floating MariaDB instance on highest-IP node, recalculated on every merge
- Always available anywhere on the mesh — same address regardless of which node hosts it
- Stores real-time sensor data as generic JSON tuples — any sensor, any type
- Intentionally ephemeral — no reconciliation on split/rejoin, matches the physics
- Network-wide shared scratchpad visible to all nodes simultaneously
- Supports real-time dashboards and AI inference simultaneously
- WellKnownSite table — service discovery without DNS, maps service names to node IPs dynamically

AI Platform

- Local Ollama inference — no cloud, no backhaul required
- Domain-specific models trained for the deployment context
- Reads live sensor data from transient database
- Writes commands back to actuators through the same database
- Sensor-to-actuator loops operate entirely offline
- Online/offline — behaves identically with or without internet
- Multiple AIs can run simultaneously across nodes, sharing the same world state
- Multiple dashboards see the same real-time data simultaneously — no master, no replica

Sensor and Actuator Platform

- ESP32 sensor and actuator boards via Bluetooth, WiFi, or Ethernet
- Pi Zero concentrators aggregate and push to local database
- External mesh radio technologies (Meshtastic, MeshCore, generic LoRa) integrate as data endpoints — their RF bridges write directly into the transient database
- FrogNet can also sit *on top of* those same mesh radio bearers — the semantic layer riding the RF network as its transport
- Any sensor type — environmental, health, telemetry, structural, agricultural
- Bidirectional actuator control validated end-to-end on April 17, 2026 (Flask UI, sensor board, actuator board, all coordinating through the transient database across a 2,400-mile WAN)
- Real-time ingestion with immediate mesh-wide visibility

Broker and Tunnel System

- StreamingFrog broker manages WireGuard tunnel lifecycle
- Broker is not restricted to any specific infrastructure
- Can run on any internet-connected server
- Can run inside a corporate VPN for fully private tunnel networks
- Can run on any node in the mesh itself
- Bidirectional tunnel management — creation, refresh, teardown
- 256-worker daemon pool for genuine parallel processing

Security Model

- Physical proximity is the first perimeter — must be in range to join
- Network membership is the trust boundary — not per-service credentials
- WireGuard encryption on all tunnels — Curve25519, compact handshake
- Inbound from internet closed by default — nodes don't advertise themselves
- No TLS overhead on internal traffic — eliminated by design, not by omission
- Standard Linux underneath — full ecosystem of security tools available if wanted
- TOFU trust model for script distribution — pins on first contact, rejects on key mismatch
- Ed25519 signatures for mesh-wide script distribution
- Not deployed on amateur (ham) radio bands — FCC Part 97.113(a)(4) prohibits obscured communications, and template-based compression is indistinguishable from a cipher to a listener without the template dictionary. This is a deliberate design constraint, not a limitation.

Wire Protocol (FNW1)

- Custom wire protocol — REQ_FULL, REPEAT, RAW, DIFF, RESP_DIFF, SAME, ERROR, SEQ_RESET, HELLO opcodes
- 4-byte big-endian length prefix on all frames
- Sequence-tagged frames for ordered delivery
- WIRE_VERSION=3, REQ_HASH_LEN=16
- LZ4 smart compression underneath BLDC-1

Management and Operations

- frognet_monitor — real-time dashboard showing all nodes, daemon status, metrics
- Admin API — X-Admin-Key protected PHP endpoint for remote node management
- Admin Android fragment — PIN-gated admin UI within FrogNet Family app
- Admin HTML dashboard — browser-based management at FrogNetAdmin vhost
- Restart individual daemons remotely — tunnel, proxy, daemon independently
- Force merge remotely — trigger sync_interfaces.sh from admin UI
- WiFi reconnection — switch antennas to new endpoints via nmcli through admin API
- frognet-runonce — Ed25519-signed script distribution across mesh via gossip relay
- Metrics pipeline — v5 sensor model, SemanticProxy, SemanticCache, SemanticDaemon sensors
- Unified logging — frognet_log.py, ERR/INFO/DEBUG via FROGNET_LOG_LEVEL

- setup_lillypad.bash — complete node provisioning script

Applications Built on the Platform

- FrogChat — vanilla JS single-file chat app, no dependencies, runs in any browser
- FrogNet Family Android app — Kotlin, GPS tracking, chat, sticky notes, dual-write architecture
- Real-time GPS presence map — all users visible mesh-wide via transient DB
- Voice and video calling — WebRTC P2P for 1:1, Galène SFU for group calls
- Family calendar — CalDAV/CardDAV, mesh-synchronized
- Shared photo gallery — self-hosted, no cloud
- Family wiki — flat-file, runs on any node
- Distributed file sync — transport-agnostic, peer-to-peer
- Wellness monitoring — passive activity signals, anomaly detection, elder care alerts
- Emergency broadcast — mesh-wide alert system, works without internet

Deployment and Distribution

- Passcode system — broker authentication for FrogNet Family tunnels
- NetworkManager fix — dns=default prevents resolv.conf overwrites
- Broker portability — any server, any VPN, any node, no vendor dependency

Intellectual Property Position

- Three issued US patents — storage virtualization, pop-up networking, physical product
- Pop-up networking patent is direct architectural precursor to mesh formation — lineage, not just a filing date
- Nine additional patentable inventions identified, four highest-priority filing defensively
- NSF Phase I SBIR invitation in hand — Wireless Technologies topic
- CAGE code 1A5Y5, SAM registered, full awards authority

Proven in Production

- Ten nodes across Seattle, New York, and Amsterdam
- First independent end-to-end closed-loop demonstration completed April 17, 2026 — Daniel Tone (New York) driving a physical actuator in New York from a Flask UI in New York, with all coordination passing through the transient database on SeattleDB, 2,400 miles away, entirely over semantic-compressed HTTP
- Sub-2-second RTT transatlantic
- 93.8% bandwidth reduction on real traffic

- 44× throughput improvement on constrained links
- Active development partner in New York City
- Community collaboration with EmergencyHam (over WiFi, LoRa, and WireGuard at EOCs — not over the amateur bands)
- CWNE #3 Keith Parsons independently invited 6-hour deep dive at Wireless LAN Professionals Conference
- Two students at King's College London onboarding as testers
- TRL 6–7

Appendix A: How FrogNet Handles Emergencies

These scenarios were referenced in the opening section and are reproduced here for readers who want concrete examples of how FrogNet performs when conventional infrastructure fails. Each one is a deployment pattern the current system supports today, not a future roadmap.

Summer Camp

Your twelve-year-old is at a camp in the mountains. There is no cell service and barely any electricity. The camp does not have commercial internet, but the ranger's station has a long-range LoRa link to a Meshtastic-style RF gateway at the trailhead, and the trailhead has a satellite uplink that comes up for a few hours each evening.

The camp has a FrogNet box. So does your house. For most of the day, the camp's box operates on its own — a local network inside the camp for the staff and kids. When the evening uplink comes up, the camp's box and yours find each other and synchronize everything that happened during the day. Your kid's check-ins, their photos, their GPS dots on the family map — all of it arrives in a single batch that the semantic compression engine sends over the narrow link in a fraction of the bandwidth it would normally take.

You leave a sticky note that says “Grandma says hi.” The camp's box picks it up the next evening. Your kid sees it when they wake up.

Nothing goes through any company's servers. No cell service is required at the camp. The link between camp and home is a few hours of narrowband a day, and FrogNet treats that as perfectly sufficient — because for this kind of family communication, it is.

The Ambulance

A paramedic crew responds to an emergency. They load the patient and begin treatment. Between the scene and the hospital, they pass through areas with no cell coverage. In a conventional system, the data they are collecting — vitals, treatment notes, patient status — cannot reach the hospital until they drive back into cell range. The emergency room gets no warning, no preparation time, no head start on the case.

With FrogNet, the ambulance has a box. The hospital has a box. The boxes connect through whatever path is available — cellular, WiFi hotspots along the route, a commercial narrowband data radio, or even a dedicated LoRa relay. Patient data flows to the hospital in real time when connectivity exists. When it does not, the ambulance's box stores everything locally and transmits the moment a link reappears. The ER sees vitals updating as the ambulance approaches. They are ready before the doors open.

This is not hypothetical architecture. FrogNet's semantic compression engine was specifically designed to deliver full web applications over links as narrow as 4800 baud — slower than a 1990s dialup modem — and has been validated at that rate. It achieves 44× the throughput of conventional protocols on the same physical link by learning the structure of the traffic and

transmitting only what has changed. On a steady-state data feed like patient vitals, a 500-kilobyte update collapses to fewer than 150 bytes on the wire. That is small enough to send over almost anything that carries IP.

The Disaster

A hurricane hits the Gulf Coast. Cell towers are down. Power is out. The internet does not exist. First responders from six agencies converge on the area with no way to coordinate because every system they use depends on infrastructure that is no longer there.

Each FrogNet box is a sovereign network. Drop one at the command post, one at the field hospital, one at the staging area, one in each search-and-rescue vehicle. They mesh automatically over WiFi when they are near each other. When they are out of WiFi range, they connect by whatever IP-capable radio is in the truck — commercial narrowband, LoRa, Part 15 ISM-band equipment, or a satellite uplink. Sensor data from the field flows into a shared database that every node on the mesh can see. An AI running on one of the boxes watches the sensor feeds and flags anomalies. Dashboards at the command post show real-time positions of every unit.

None of this requires the internet. None of it requires a cell tower. None of it requires any infrastructure that the hurricane could take out. The FrogNet is the infrastructure, and it came in the back of a truck.

The network grows. When two independent FrogNets come within range of each other, they discover each other and merge into a single, larger network — automatically, with no human intervention. A rescue team from Texas arrives with their own FrogNet. They drive within WiFi range of the command post. The two networks see each other, exchange routing information, and become one. Every node on both sides can now reach every other node. This happens in seconds, without anyone configuring anything.

The same merging works across different transport types simultaneously. The command post connects to the field hospital by Ethernet. The field hospital connects to a remote unit by narrowband radio. A supply convoy connects through a satellite uplink. From the application's perspective, it is all one network. The applications do not know and do not care whether the bytes travel by wire, WiFi, radio, or satellite. FrogNet handles the transport. The applications just work.

Small Business, Farm, and Other Everyday Uses

A small business with three locations and ten remote employees replaces Google Workspace with FrogNet boxes at each office and each home. Internal files, messaging, a company wiki, and sensor monitoring at the warehouse — all running on their own hardware, all private, no monthly per-seat fees, no corporate data passing through anyone else's servers.

A family farm deploys sensor nodes in the fields and a FrogNet box at the farmhouse. Soil moisture, temperature, humidity — all flowing into a local dashboard through a LoRa or Meshtastic RF bridge. A local AI makes irrigation recommendations. None of this requires cell coverage in the field. None of it requires a cloud subscription to view your own sensor data.

Fawcett Innovations LLC · unixpro@ufie.org · (206) 335-9639 · CAGE 1A5Y5