


# Transit Fabric: A Cryptographically Verifiable fc00::/8 eBGP Transit Fabric and Multi-Commodity Routing Market on Bittensor

## Contents

What changed in version 1.3
What changed in version 1.2
What changed in version 1.1
Abstract
1. Introduction & Motivation
2. System Architecture & Cryptographic Identity
3. Cryptographic Accounting & Conservation of Flow
4. Dual-Mechanism Incentive Design
5. Prosumer Edge Fleet Aggregation & The "Mine-to-Surf" Loop (`ADR-0019`) — later phase
6. Tokenomics: Emissions, Wholesale and the Buyer-Side Burn
7. BandwidthOS, the Model Home: Commercial Operations & Go-To-Market Workflows
8. Security, Privacy & Wire-Obscurity Analysis
9. Empirical Evaluation & Multi-Region Benchmarks
10. Normative Protocol Parameters
11. Conclusion

> Taken from the Transit Fabric source repository at commit 10a682c (10a682c933278f64f47f65e25711cd66d74552ac, main), file docs/whitepaper.md. Links into the private source tree are shown as plain text.

**Version:** 1.3 (2026-11-13). Replaces 1.2, published earlier the same day.

**Status labels.** Every feature in this paper carries one of three labels:

- **Live:** runs today on Bittensor testnet, subnet 611 (tester build `v0.1.0-tester.3`). Validators on 611 are all run by the Transit Fabric team for now.
- **Designed:** specified in an ADR or spec, and built or partly built, but not yet on, or not yet scored. A design whose ADR is still *proposed* says so.
- **Planned:** decided or approved, and not built yet. A **later phase** is planned work off the launch path.

Launch is transit (Mechanism 0) and route solving (Mechanism 1) only. Transit Fabric is not yet registered on mainnet, so nothing described here runs there.

## What changed in version 1.3

These changes follow the founder's answers to the 1.2 review on 2026-11-13.

1. **Economics rewritten to the current model** (§6; ADR-0053, ADR-0063, ADR-0064, ADR-0065). Removed: the 1.1 model's 80/20 company buyback, the company's 20% fee on protocol revenue, the `ALPHA` step-down tied to it, the subsidy ratios, and the three-year revenue, market-cap and company-income projections. §6 now says who pays whom, in what unit and what is burned, with ADR-0063's rate-card and clearing formulas, ADR-0065's TAO/USD conversion and a worked example, and the burn guards' values from the OQ-291 calibration. No new projection replaces the old ones.
2. **How BandwidthOS earns** (§6.3, §7). A reseller's margin, never protocol revenue. The protocol holds no money and charges no fee.
3. **BandwidthOS may run nodes, as a reference deployment** (founder, 2026-11-13; ADR-0064 §8; §7.2). Open-source code and a public example for others to copy, with no protocol privilege and no special role. The hub-node projection and every "anchor" role are removed. How these nodes' emissions sit with the company's treasury rule is open (OQ-301).
4. **Go-to-market sequences** (§7.4). The contract figures stay. The 1.1 buyback figure and the capacity-pay projection in Sequence 1 are removed; the BFD failover change in §7.1 stays.
5. **Change notes, abstract and §4.1** brought in line: 1.1 item 7 and 1.2 items 3 and 5 point to the new §6 and §7.2, and §4.1's step 5 no longer describes the `ALPHA` step-down as current. The company form is still not decided (Inc. or LLC).
6. **Owner share, BandwidthOS's nodes and telemetry** (founder, 2026-11-13; ADR-0064 §8, OQ-301 closed). The subnet owner's 18% emission share is the owner's own, not BandwidthOS revenue. BandwidthOS's nodes are a reference example: their pay is ordinary operator pay, BandwidthOS burns TAO like every reseller, and its node UIDs are not separately disclosed. The telemetry feed is a later phase.

## What changed in version 1.2

These decisions were made between 2026-11-08 and 2026-11-13 and are reflected in the sections below. Where a section still describes an older design, it says so.

1. **Route solvers also discover peers: the peering plan** (System One autonomous peering design, approved 2026-11-12; abstract, §2.3, §4.2.1). Solvers compete on a peering plan, the adjacency set for the next window, alongside their route tables, and validators score plans and attest the winner by quorum. Node and solver RTTs are hints that can only narrow a node's placement. **Planned, not built:** its two ADRs and an incentive red team come first. Today peering is self-service.

2. **System One on the launch path** (ADR-0067; §2.4). `fabric-sense` ships first on kernel, no-root and container nodes, as rules only. Two jobs are live on testnet 611: rebuilding a stale validator-set anchor, and automatic fault reports that are never a scoring input. `bwos doctor`, `bwos report` and an agent skill share one symptom registry. The model, the skill library and routers come later. This narrows 1.1 item 6.
3. **The buyer burns; nobody holds the money** (ADR-0063, proposed, revision 5; the founder chose burn on 2026-11-11; §6.2, §7.3). A reseller buys wholesale capacity by calling `add_stake_burn`, which buys the subnet's alpha from the pool and destroys it in the same call, signed by the buyer's own coldkey. There is no protocol treasury and no custodian. This replaces 1.1 item 2's "the company converts it to TAO and buys alpha" and the 1.1 80/20 buyback; 1.3 removes the projections built on it (§6). Details pending red team and simulation; not built (PB-49).
4. **Wholesale is quoted in US dollars and settled by burning TAO** (ADR-0065, accepted in principle on 2026-11-12, details proposed pending a red team; §6.2, §7.3). Each window's rate card carries one TAO/USD reference read from the chain at a fixed block. Bittensor's Gamma credits are a later option, not part of v0.
5. **BandwidthOS is the "model home"** (ADR-0064; §7). BandwidthOS is the one company around Transit Fabric: the first reseller and first gateway. Everything it runs is open source, it has no protocol privilege, and it pays the same clearing price as any reseller. Its own alpha comes only from retail margin and sits on hotkeys that never set weights. This replaces 1.1 item 7's framing and §7.2's company-owned anchor routers with special roles. *(1.3: BandwidthOS may run nodes only as a reference deployment, §7.2.)*
6. **The name is Transit Fabric** (ADR-0062, ADR-0066; title and throughout). Transit Fabric is the open protocol and network; BandwidthOS is only the company. Code keeps the `bwos` names until a trademark search before mainnet. The project's old name is retired everywhere.
7. **Inline node bindings** (ADR-0058; §2.1). Live. A hotkey binds up to 7 nodes inside its own on-chain commitment, with no server to publish from.
8. **Linking with a browser wallet** (ADR-0054; §2.1). Live. A static page lets Talisman, SubWallet or Polkadot.js sign the binding, so no hotkey file sits on the operator's machine.
9. **Probe sessions keyed by the node key** (ADR-0061; §2.1). Live. A node's probe sessions use the X25519 form of its node key, so no operator hands WireGuard bindings to validators.
10. **Proof of measurement work** (ADR-0055, proposed, scoring gated off; §4.1). Validators would be paid a capped slice for measurement bytes nodes signed for, so measuring pays better than copying.
11. **Peer-load capacity tests** (ADR-0056, proposed, scoring gated off; §3.3). Other operators' nodes make the load, with hidden validator slices, about 3 random calls a day with a reputation bond. If adopted, this replaces 1.1 item 3's "about 20 full-rate tests a day" and ADR-0039's validator-pushed team tests.
12. **Locational scarcity pricing** (ADR-0057, proposed, gated off; §4.1). Capacity would pay more where it is scarce relative to demand, with placement from validator RTTs that a node can only make look farther, never closer.
13. **Sharded solving** (ADR-0059, proposed, gated off; §4.2.2). Solving split by traffic class and destination region, with a fixed published merge of shard winners.
14. **Launch scope** (`docs/launch-stages.md`; §5, §7, §8.2). VPN exits, "Mine-to-Surf", fleet aggregator vouchers and the Transit Fabric Edge appliance are later phases, off the launch path. Registered gateways (ADR-0052) are planned before mainnet. §8.2 no longer claims node links resist DPI (ADR-0040).

## What changed in version 1.1

These decisions were made on 2026-11-06 and are reflected in the sections below. Where a section still describes the older design, it says so. Version 1.2 revises some of them: item 2 by 1.2 items 3 and 4, item 3 by item 11 (proposed), item 4 by items 3 and 5, item 6 by item 2, and item 7 by item 5. Version 1.3 removes the projections built on item 2's buyback (§6).

1. **Unearned weight goes to a disclosed first-party UID, not a burn or recycle UID** (§4.2; `scoring.md` §Unearned weight and the first-party UID). Version 1.0 sent the reference solver's share to a burn UID. That was dropped: the usual way to burn on Bittensor is to weight the owner's own hotkey, which the chain penalises by cutting the subnet's TAO share.

    - **Routing (Mechanism 1):** unearned weight goes to a public first-party UID on a separate coldkey, and the amount is reported every window. Unearned weight means the reference solver's share, or everything when no solver scores. It shrinks as outside solvers beat the reference solver by more than `EPSILON_ROUTE` (2%).
    - **Transit (Mechanism 0):** unearned weight goes to no one. If no node has proven capacity in a window, validators set no transit weights at all.
2. **Nodes are paid only in emissions, and customer traffic is checked, never credited** (ADR-0053; §3.2, §4.1). Customers pay the company in USDC. The company converts it to TAO and buys alpha. *(Revised in 1.2: no company sits in the middle. A reseller burns TAO for wholesale capacity with* `add_stake_burn`*, which buys alpha and destroys it, ADR-0063 and ADR-0065.)* Vouchers, payments and client receipts are commercial metering, never a scoring input, so faking or steering "paid" traffic earns nothing. Mishandling customer traffic can only lower a node's score.
3. **An always-on canary stream plus random full-rate bursts proves transit** (ADR-0051; §3.3). Validators keep a low-rate canary stream through every node, on routes they allocate, checked at each hop. Frequent random full-rate tests catch burst-rented capacity.
4. **Buyback alpha never becomes validator power** (ADR-0053, OQ-105). The company holds the alpha it buys on hotkeys that never set weights, and that stake counts for nothing in measurement. *(Revised in 1.2: wholesale burns leave no alpha to hold. BandwidthOS's own alpha comes only from retail margin, under the same rule, ADR-0064 §6.)*
5. **Gateways** (ADR-0052; §7). Registered datacenter operators can on-ramp customers over BGP, cross-connects or a proxy. They can off-ramp to a customer's own network, or to allowlisted public destinations, so the customer's far end needs no Transit Fabric software. Ordinary transit nodes still never egress.
6. **One System One sidecar on every node, with a public skill library learned from the fleet** (ADR-0047, ADR-0048, ADR-0050).
7. **The operating company is BandwidthOS** (Inc. or LLC, not yet decided). *(1.3: it earns a reseller's margin, never protocol revenue, §6.3.)*

**Code:** a Rust workspace (`bwos-core`, `bwos-net`, `bwos-node`, `bwos-solver`, `bwos-validator`, `bwos-probe`, `bwos-chain`, `bwos-cli`, `bwos-client`, `bwos-sim`). Code identifiers keep the `bwos` prefix until the trademark search before mainnet (ADR-0062, ADR-0066). Transit Fabric is the protocol and the network; BandwidthOS is a separate company, its first reseller (ADR-0064).

---

## Abstract

The global internet backbone suffers from two structural inefficiencies: **(1) stranded physical capacity**, where datacenter and hosting operators purchase committed 95th-percentile transit and leave 30–60% of their committed headroom idle, and **(2) performance-blind routing**, where the Border Gateway Protocol (BGP) selects paths by Autonomous System (AS) hop count and bilateral billing policy rather than real-time latency, jitter, packet loss, or link saturation. Hyperscalers (Google B4, Meta Express Backbone, Cloudflare Argo) solved these problems internally using private Software-Defined Wide Area Networks (SD-WANs) and centralized Multi-Commodity Flow (MCF) traffic-engineering solvers, but those capabilities remain locked inside proprietary corporate backbones. Meanwhile, first-generation decentralized bandwidth and VPN networks failed to capture institutional datacenter supply due to exit-node abuse liability, trivial self-dealing (wash-trading) of byte counters, and userspace cryptographic bottlenecks.

We present **Transit Fabric**, a Bittensor subnet that turns committed datacenter and prosumer edge bandwidth into an open, cryptographically verifiable Layer-3/Layer-4 transit fabric. Every node and client in Transit Fabric derives a deterministic, unforgeable `/128` IPv6 Unique Local Address (`fc00::/8`) directly from its Ed25519 public key, forming a flat, zero-registry eBGP overlay (`ASN 4200000000+`) over WireGuard links: kernel WireGuard on kernel nodes, and userspace WireGuard on nodes that run without root. IPsec ESP (`RFC 4303`) and IEEE 802.1AE MACsec backbone links are planned for later. To align incentives without a central coordinator, Transit Fabric splits Bittensor emissions across two specialized mechanisms: **Mechanism 0 (`mechid=0`)** rewards transit operators for verified bottleneck capacity and conservation-of-flow traffic delivery, while **Mechanism 1 (`mechid=1`)** runs a continuous tournament among route solvers that compute capacity-aware, multi-class (`realtime`, `low_latency_hd`, `bulk`, `economy`, `resilient`) route tables. Solvers will also discover peers: in the planned design, they

compete on a **peering plan**, the set of links the fabric runs next window, which validators score and attest by quorum, so no node or company chooses the topology. Transit nodes never egress to the public internet; only registered exit partners and registered gateways do (`ADR-0004`, `ADR-0052`). Nodes are paid only in emissions: on testnet today for uptime and capacity proven by validator tests, and, once the designed canary stream is scored, for carriage of a validator-allocated canary stream checked at every hop (`ADR-0051`). Customer traffic is checked, never credited (`ADR-0053`), so wash-trading earns nothing, while forwarding stays at kernel line rate. Customer money reaches the subnet by a burn the buyer makes itself: a reseller buys wholesale capacity, quoted in US dollars, by burning TAO into the subnet's pool, so no treasury or company holds it and the protocol charges no fee (`ADR-0063`, proposed; `ADR-0065`). BandwidthOS, the first reseller, earns only its retail margin (`ADR-0064`). We validate the protocol through a 50-window adversarial game-theoretic simulator, live transatlantic multi-region deployments across North America and Europe, and a public test network on Bittensor testnet, subnet 611.

---

## 1. Introduction & Motivation

### 1.1 The Stranded Datacenter Transit Opportunity

Institutional datacenter operators, bare-metal cloud providers, and regional colocation facilities purchase upstream IP transit on fixed monthly commitments (typically billed at the 95th percentile of 5-minute traffic samples over 10 Gbps, 40 Gbps, or 100 Gbps ports). To survive diurnal traffic peaks and DDoS bursts without incurring punitive overage rates, operators provision substantial headroom that sits idle for the majority of the billing cycle. Because the commit is already sunk, the marginal cost to an operator of carrying additional overlay transit traffic below their 95th-percentile commit is near zero.

### 1.2 The Pathology of Public Internet BGP

Even when physical fiber paths between two regions are uncongested, public internet BGP routinely selects suboptimal routes:

- **AS-Path Blindness:** Standard BGP selects routes primarily by `LOCAL_PREF` and `AS_PATH` length. A 2-hop AS path with 4% packet loss and 45 ms of queuing jitter is preferred over a 4-hop AS path with 0% loss and 12 ms lower propagation delay.
- **Slow Convergence:** Public BGP route convergence following a fiber cut or peering brownout can take tens of seconds to minutes, disrupting real-time voice, video, gaming, and distributed AI training workloads.
- **No Native Quality-of-Service (QoS) Differentiation:** Public BGP cannot simultaneously route a latency-critical flow over a low-jitter path, a bulk dataset transfer over a widest-bandwidth path, and a high-availability flow over interior-node-disjoint primary and backup paths.

Hyperscalers solved this by treating all of their datacenters as nodes in a single flat graph controlled by a traffic-engineering solver (such as Google's B4 and Cloudflare's Magic WAN). Transit Fabric brings this exact architecture to an open, permissionless market of independent operators.

### 1.3 Why First-Generation Decentralized Bandwidth Networks Failed

Prior attempts at tokenized bandwidth markets and decentralized VPNs (dVPNs) encountered three fatal failure modes:

1. **The Exit Liability Trap:** Legacy dVPNs required every bandwidth provider to act as a public internet exit node. When an anonymous user routed abusive traffic (DMCA violations, credential stuffing, or port scanning) through a node, the hosting operator received abuse complaints and risked immediate upstream BGP session termination. Consequently, institutional datacenter operators refused to participate, leaving legacy dVPNs with unreliable residential proxies.
2. **The Wash-Trading Death Spiral:** Protocols that rewarded miners proportional to "bytes relayed" without route-assignment enforcement or burn-backed demand verification were immediately gamed. Miners spun up two colluding nodes in the same datacenter rack and transmitted gigabits per second of synthetic loop traffic to capture 99% of token emissions.
3. **Userspace Mixnet Bottlenecks:** Privacy mixnets that wrap every packet in multi-layer userspace cryptography (e.g., Sphinx packet formats) incur severe CPU and serialization overhead, reducing throughput to a few megabits per second and adding hundreds of milliseconds of latency—rendering them unusable for institutional transit or daily desktop

networking.

Transit Fabric resolves all three failure modes at the protocol layer:

- **Strict Transit vs. Exit Separation (`ADR-0004`):** Mechanism 0 transit nodes **never** egress packets to the public internet; they only forward encrypted `fc00::/8`-to-`fc00::/8` overlay traffic between bound peers. Public internet egress is performed only by registered **Exit Partners** (crypto-accepting VPN providers) and **registered gateways** (datacenter operators that meet published criteria, carrying paid, allowlisted traffic only, `ADR-0052`). Neither exists yet: gateways are planned before mainnet, and VPN exits are a later phase. On testnet 611 nothing leaves the overlay.
- **Solver-Assigned Routes + Conservation-of-Flow Receipts:** Miners cannot choose their own traffic paths. Transit is only rewarded on paths assigned by the winning Mechanism 1 route table (or unannounced Validator Canary Probes), verified by adjacent co-signed `HopReceipts` and per-hop digests. Since version 1.1, nodes are paid only in emissions, and customer traffic is never credited (`ADR-0053`), so synthetic loop traffic earns nothing.
- **Cryptographic Control Plane, Kernel Data Plane:** Cryptography is used to derive `/128` IPv6 identities, authenticate route tables, and co-sign periodic byte-counter receipts, while packet forwarding runs at line rate in the OS kernel (`WireGuard` and `FQ/BBR` today; `IPsec ESP` and `MACsec` planned). Nodes without root forward in userspace instead, at lower rates (§2.3).

---

## 2. System Architecture & Cryptographic Identity

The diagram shows the launch path (Mechanism 0, Mechanism 1 and validators) and, dashed, the later-phase client and exit layers.

```
flowchart LR
    subgraph Edge["Client & Edge Layer (later phase)"]
        C["bwos-client (macOS / Windows / OpenWrt)\nEd25519 Key -> fc00::/8 (/128)"]
    end

    subgraph Mech0["Mechanism 0: Transit Fabric (fc00::/8 eBGP over WireGuard)"]
        I["Ingress Node\n(WireGuard + HopReceipt)"]
        T1["Transit Node A\n(kernel or no-root, eBGP + System One)"]
        T2["Transit Node B\n(kernel or no-root, eBGP + System One)"]
    end

    subgraph ExitLayer["Exit Partners & Gateways (ADR-0004, ADR-0052; not on testnet)"]
        E["Registered Exit or Gateway\n(egress for paid, allowlisted traffic)"]
        W["Public Internet\n(TLS 1.3 / TCP / UDP)"]
    end

    subgraph Control["Validators & Mechanism 1 Solvers"]
        S["Route Solvers (mechid=1)\nRoute tables; peering plans (planned)"]
        V["Validators (bwos-validator + bwos-probe)\nProbe Sessions + Capacity Tests + Subtensor Weights"]
    end

    C -. "1. Client traffic (later phase)" .-> I
    I -- "2. eBGP Assigned Path\n+ Co-Signed HopReceipt" --> T1
    T1 -- "3. eBGP Assigned Path\n+ Co-Signed HopReceipt" --> T2
    T2 -. "4. Co-Signed HopReceipt" .-> E
    E -. "5. Public Egress (:443)" .-> W
    S -- "Commit-Reveal RouteTable\n(+ PeeringPlan, planned)" --> V
    V -- "Attested RouteTable\n(+ PeeringPlan, planned)" --> Mech0
    V -- "Probes and Capacity Tests" --> Mech0
```

### 2.1 Separation of Network Identity and Economic Identity

Transit Fabric enforces a strict cryptographic boundary between a machine's **network identity** and an operator's **economic identity** (`docs/specs/identity.md`):

| Key Type | Algorithm | Storage Location | Protocol Role |
|---|---|---|---|
| **Node Key** | `Ed25519` (RFC 8032) | Node / Appliance (`~/.bwos/node.key`, `0600`) | Network identity; derives `fc00::/8` IPv6 address; signs `HopReceipts`, `WgBindings`, the node's consent and its fault reports. Its X25519 form keys the node's probe sessions (`ADR-0061`) |
| **WireGuard Key** | `X25519` (RFC 7748) | Node / Client (`~/.bwos/wg.key`, `0600`) | Ephemeral/rotatable tunnel transport key bound via signed `WgBinding` |
| **Operator Hotkey** | `sr25519` (Subtensor) | Off-box Operator Wallet (browser wallet or CLI) | Economic identity (UID); signs `NodeBindings` and `CapacityDeclarations`, or commits an inline binding |
| **Client Key** | `Ed25519` (RFC 8032) | End-user device (`~/.bwos/client.key`, `0600`) | Later phase. Derives client `fc00::/8` address; signs `ClientReceipts` and `FleetEnrollments` |

Appliances and transit servers **never hold a Bittensor hotkey**, and System One never reads a key or signs anything (Invariant 5, `ADR-0067`). Each node signs a **consent** naming the hotkey it may be bound to, and delivers it to validators over its own probe session. An operator then links nodes to their Bittensor UID in one of two ways:

- **Inline binding (live, `ADR-0058`).** For 1 to 7 nodes, the hotkey writes the node keys and a validity range straight into its own on-chain commitment, in at most two 128-byte fields. The chain already authenticates the hotkey and subnet, so no signature and no server are needed. An inline binding takes precedence over any other binding the UID names.
- **Manifest binding (live, `ADR-0015`, `ADR-0036`).** For more nodes, the operator publishes a hotkey-signed `NodeBinding` (domain `node-binding`), whose SHA-256 digest is committed on-chain through a commitment manifest and served via content-addressed HTTP (`/v0/objects/node-binding/<sha256>`).

Either way, the operator can sign with a browser wallet (Talisman, SubWallet or Polkadot.js) on a static "Link with your wallet" page, with no hotkey file on their machine (live, `ADR-0054`), or with the `bwos bind` command. A node counts only in a binding whose hotkey it has consented to, so listing someone else's node gains nothing. A node that has consented to two hotkeys whose bindings both survive counts for neither (`docs/specs/identity.md`, Binding resolution).

### 2.2 Zero-Registry Cryptographic IPv6 Addressing (`fc00::/8`)

On the public internet, participating in BGP requires acquiring an Autonomous System Number (ASN) and IP prefix from a Regional Internet Registry (ARIN, RIPE, APNIC) and maintaining RPKI Route Origin Authorizations (ROAs).

In Transit Fabric, every node and client deterministically derives a unique `/128` IPv6 address inside the RFC 4193 Unique Local Address (`fc00::/8`) space directly from its 32-byte Ed25519 public key pk:

$$\text{digest} = \text{SHA-512}\big(\text{SHA-512}(\text{pk})\big)$$

$$\text{addr}_{128} = \texttt{0xfc} \;\|\; \text{digest}[0..15]$$

This construction provides three critical properties:

1. **Zero Key Grinding:** Unlike CJDNS (which requires brute-force CPU grinding until a double-SHA-512 hash begins with `0xfc`), Transit Fabric prepends the `0xfc` ULA prefix byte to the first 15 bytes (120 bits) of the double-SHA-512 digest. Key generation takes less than 50 microseconds on any CPU while providing $2^{120}$ collision resistance.
2. **Impossible BGP Prefix Hijacking:** Because the `/128` IPv6 address is a one-way cryptographic function of pk, no node on the overlay can originate or spoof a route for `fcxx:...` without possessing the corresponding Ed25519 private key.
3. **Universal Peer Parity:** Every participant—whether a 100 Gbps core router in Frankfurt, a cloud VM in Illinois, or a macOS laptop on Wi-Fi—speaks the exact same `fc00::/8` addressing scheme.

### 2.3 Flattened eBGP Control Plane and Kernel Data Plane

Rather than inventing a custom routing protocol, Transit Fabric treats every node as a first-class datacenter router speaking standard eBGP and BFD over point-to-point encrypted links. Two node kinds speak the same protocol, and both are live on testnet 611 (`ADR-0033`):

- **Kernel node:** Linux with root, kernel WireGuard and **FRRouting (`bgpd + bfdd`)**, set up by one install script.
- **No-root node:** one static binary with no root, no TUN device and no capabilities, as a process or a container. It runs BGP, BFD and WireGuard (`boringtun`) in userspace on an in-process IPv6/TCP stack. One flow over it carries about 220 Mb/s on cloud VMs.

Both kinds share these properties:

- **Deterministic 4-Byte Private ASN Derivation:** Each node derives an RFC 6996 4-byte private ASN from its public key (`OVERLAY_BASE_ASN = 4,200,000,000` + deterministic node offset), originating its `/128` `fc00::/8` loopbacks into eBGP.
- **Sub-Second BFD Failover:** Bidirectional Forwarding Detection (`bfdd` on kernel nodes, `bwos_net::bfd` on no-root nodes, `3 × 200 ms` intervals, so 600 ms detection) monitors every adjacent peer link. If a primary link drops, the BGP session ends and traffic moves to the solver-precomputed `backup_path`: measured at 544-716 ms in namespaces and an outage of 520-680 ms over the internet (`docs/results/phase-a-spikes.md`), without waiting for global control-plane reconvergence.
- **Transport Encryption & Host Kernel Tuning:**
  - **Every node today (live):** WireGuard (`ChaCha20-Poly1305`) with key-rotated `WgBindings`: kernel WireGuard on kernel nodes, userspace WireGuard on no-root nodes.
  - **Datacenter Backbone (planned, `ADR-0010`):** Point-to-point **IPsec ESP (`RFC 4303`, AES-256-GCM)** or Layer-2 **IEEE 802.1AE MACsec (`GCM-AES-128/256`)** with deterministic pairwise key derivation (`ipsec-ike-psk`, `ipsec-esp-key`, `macsec-cak`, `macsec-ckn`) for hardware NIC crypto offload (`mlx5` / Intel QAT).
  - **Host TCP/UDP Offload Profile (`bwos-node tune-host`):** Configures Linux `fq` pacing + `BBR` congestion control, 64 MB socket buffers (`net.core.rmem_max = 67108864`), and `ethtool` UDP GRO forwarding (`rx-udp-gro-forwarding on`).

#### Peering: who links to whom

A node is paid for capacity only when it has a data plane: a BGP session with at least one node of another UID (live). Which nodes peer is therefore part of the network's design, not each node's private choice.

- **Today (live, self-service).** An operator posts a peer card in a "Peering request" Issue. A public node the team runs adds the new node by itself (autopeer) once validators list it, within per-UID and total caps. Operators can also swap peer cards with each other. Two nodes both behind NAT can't peer yet.

- **Planned: peer discovery by System One and the solvers** (System One autonomous peering design, approved 2026-11-12; not built). The goal is that a new node, checked an hour after it joins, has found good, diverse peers by a draw no one can steer, and is carrying traffic.

  1. **Publish.** Each node publishes a signed peering record (its WireGuard key, its reachable endpoints or "dial-out only", its `WgBinding`), which validators carry in the attested directory.
  2. **Draw.** Every window, each node gets candidates from a draw seeded by `SHA-256(block hash at the window's start ‖ node key)`, which validators can recompute. Diversity is a constraint of the draw, not a preference: no two candidates share an operator, an ASN or address block, or, where others exist, the node's own region.
  3. **Bootstrap and measure.** A new node peers with its drawn candidates at once, within each side's caps, and measures every link.
  4. **Plan.** Route solvers compete to produce the **peering plan** for the next window, and validators attest the winner by quorum (§4.2.1). Nodes apply the attested plan with hysteresis and a cap on changes per window. Every plan keeps a share of each node's links from its current draw, so new nodes are always discovered.

5. **Suspend, with evidence.** A node may suspend a link that is failing right now, for a bounded time, and publishes the measurement as evidence. Whether the link is dropped is the next plan's decision, from measurements that several reporters corroborate.

Neither the subnet owner nor BandwidthOS gets any peering advantage (`ADR-0064` §4).

### 2.4 System One: the node's own watchdog (`ADR-0067`)

**System One** (`fabric-sense`) is signed code that runs beside every node and acts in the fabric's interest by following rules anyone can check. It holds no key and signs nothing; where a step needs a key or a human, it says exactly what to do.

- **Rules and model.** Rules decide everything that affects other nodes, deterministically. A model may diagnose locally and recommend, but only rules act. At launch System One is **rules only**; the model comes later (`ADR-0047`, `ADR-0048`).
- **Three outcomes.** Every action ends as `fixed`, `recommend` or `report`, tied to a code in one **symptom registry** (`docs/specs/symptoms.md`) that `bwos doctor`, System One and agents all read. A fix runs at most 3 times a day per code, then escalates instead of looping.
- **Live on testnet 611,** on kernel, no-root and container nodes (routers later):
  - **Job 2: rebuild a stale validator-set anchor.** When the node's validator set expires, System One builds a current one from the chain and verifies it. In a container it writes the new set in place; on a kernel node it only offers a candidate, which the operator adopts with one command, under quorum-overlap checks, because the validator set is the node's trust root.
  - **Job 5: automatic fault reports.** A crash, or a problem System One could not fix, becomes a signed fault report with a fixed set of fields (code, version, commit, node kind, platform, an assertion hash, window, and the fix tried). It carries no logs, addresses, keys or peers. The node's probe session signs and forwards it to validators, at most 4 per node per window. Validators publish counts per code and version only. The operator controls what is sent (`report.auto`), and fault reports are **never a scoring input**.
  - `bwos doctor` lists a node's checks as JSON and changes nothing; `bwos report` builds a redacted bug bundle; an **agent skill** lets an AI agent install, check and troubleshoot a node under hard rules (testnet only, never a coldkey, seed or hotkey file).
- **Planned:** automatic peering (job 1, §2.3), capacity-declaration lapse notices (job 3), NAT and closed-port fixes (job 4), the model in report-only mode, and the public skill library learned from the fleet (`ADR-0050`).

---

## 3. Cryptographic Accounting & Conservation of Flow

A decentralized transit market is only as strong as its resistance to fabricated traffic. Transit Fabric enforces **conservation of flow** across every multi-hop path using four cryptographic primitives serialized in strict canonical JSON (`docs/specs/encoding.md`: lexicographically sorted UTF-8 keys, no whitespace, integer-only numbers, and domain-separated signatures over `"BWOS-v0/" || domain || "\0" || canonical_bytes`).

### 3.1 Adjacent Co-Signed `HopReceipts`

Every `RECEIPT_INTERVAL` (`30 s`), each pair of adjacent nodes $(u, d)$ on an active path exchanges and co-signs cumulative byte counters for the current window $w$ across three accounting classes (`paid`, `free`, `canary`):

```
{
  "v": 0,
  "window": 1234,
  "seq": 17,
  "up": "<upstream_node_pubkey_hex>",
  "down": "<downstream_node_pubkey_hex>",
  "bytes": { "paid": 123456789, "free": 0, "canary": 4567890 },
  "sig_up": "<ed25519_hex>",
  "sig_down": "<ed25519_hex>"
}
```

1. **Monotonicity:** Within a window $w$, byte counters for each class MUST be monotonically non-decreasing with sequence number `seq`.
2. **Bilateral Tolerance Gate:** Upstream node $u$ proposes its cumulative transmitted bytes and signs `sig_up` (domain `hop-receipt`). Downstream node $d$ compares $u$'s proposed counts against its own received byte counters. If the discrepancy is within `RECEIPT_TOLERANCE_PPM` ($5,000$ ppm $= 0.5\%$), $d$ appends `sig_down`. Otherwise, $d$ counter-proposes its own lower counts. Unilateral (single-signed) receipts earn zero credit.
3. **Multi-Hop Chain Conservation Check:** Given an assigned route $n_0 \to n_1 \to n_2 \to \ldots \to n_k$, validators verify the chain of highest-`seq` co-signed receipts $(n_0, n_1), (n_1, n_2), \ldots, (n_{k-1}, n_k)$:
   - If any interior hop $n_i$'s reported bytes exceed both its upstream receipt $(n_{i-1}, n_i)$ and downstream receipt $(n_i, n_{i+1})$ by more than `RECEIPT_TOLERANCE_PPM`, hop $n_i$ is flagged for inflation and its credit on that path is **zeroed**.
   - If any interior receipt $(n_{i-1}, n_i)$ is missing, the entire multi-hop chain is **zeroed** for that window segment.
   - Across diamond or DAG topologies where multiple routes traverse a shared bottleneck link $(n_a, n_b)$, validators aggregate flow conservation across the union of assigned routes so legitimate shared hops are never falsely penalized (`crates/bwos-core/src/validator.rs`).

### 3.2 Boundary Attestation: `ClientReceipts` and `BandwidthVouchers`

> **Superseded for scoring by ADR-0053 (version 1.1).** Client receipts and vouchers remain commercial metering: quota, billing, and destination allowlists for gateways. They are never a scoring input. A colluding path gains nothing by co-signing fake `paid` bytes, because paid bytes earn no emissions. The 1.0 text below describes the retired design and is kept for reference. Its code paths are removed in roadmap PB-40.

While `HopReceipts` prevent interior nodes from inflating counters relative to their neighbors, what stops an entire colluding path $n_0 \to n_1 \to n_2$ from co-signing fake `paid` bytes?

Transit Fabric closes this boundary loop with two end-to-end mechanisms (`ADR-0016`, `ADR-0019`):

1. **Client Receipts (`client-receipt`):** The end-user client (`bwos-client`) signs cumulative delivery receipts binding `(window, client, ingress, egress, bytes)`. Validators verify that `paid` bytes claimed at the ingress hop $(n_0, n_1)$ are covered by valid `ClientReceipts` within `RECEIPT_TOLERANCE_PPM`. Any uncovered `paid` bytes are automatically downgraded to `free` (`W_FREE = 250,000` ppm vs. `W_PAID = 1,000,000` ppm).
2. **Burn-Backed Bandwidth Vouchers (`bandwidth-voucher`):** To qualify as `paid` traffic (`W_PAID`), a session must be authorized by a `BandwidthVoucher` signed by a registered Exit Partner or Fleet Aggregator:
   - Issuing a `BandwidthVoucher` requires a mandatory minimum protocol burn/fee of `PROTOCOL_VOUCHER_BURN_PPM` ($50,000$ ppm $= 5\%$, with 10% default in simulation).
   - Because any miner attempting to mint vouchers to wash-trade their own transit nodes forfeits the protocol burn upfront and must share Mechanism 0 emissions with all honest capacity and usage on the subnet, self-dealing is strictly negative-EV.

### 3.3 Unannounced Canary Probes & Random Saturation Capacity Tests

> **Updated in version 1.1 (ADR-0051, ADR-0053):**
>
> - **An always-on canary stream.** Validators keep a low-rate canary stream through every node, on node pairs they allocate by a verifiable draw. The production table supplies only the hops.
> - **Secret schedules.** Each validator's schedule is secret until the window ends: it is committed before the window and revealed after.
> - **Checked at every hop.** Only bytes that reach the validator's sink and pass per-hop digests count.
> - **Shaped like real traffic.** Canaries follow global traffic statistics and include full-rate flows, so throttling shows up as well as dropping.
> - **Frequent random full-rate tests.** About 20 a day per node on testnet (`CAP_TEST_PERIOD` 360 blocks, Task 15b), so capacity rented only for test bursts shows up on the renter's bill.
> - **Team tests.** The full-rate tests run 60 s and are pushed by a team of validators (ADR-0039).
>
> Specified in Task 14 and PB-38. The 1.0 text below describes the earlier shared, computable schedule.

> **Status in version 1.2:**
>
> - **Live on testnet 611.** Each node opens a probe session to every validator's probe agent. Validators check over it that the node answers (uptime), relay link probes through the node to its BGP peers, and test its capacity: a public node runs a capacity-test sink, and a node behind NAT dials out when a validator asks (an interim method that credits at most 100 Mb/s).
> - **Designed, not yet scored.** The always-on canary stream checked at every hop (ADR-0051, PB-38).
> - **Proposed, scoring gated off: peer-load capacity tests (ADR-0056).** Instead of validators pushing the load, other operators' nodes make it. A test of node `N` is cut into slices that run from a source node, through `N`, to a sink, each endpoint drawn uniformly per UID from other operators by a coordinator's committed secret, and never sharing `N`'s UID, coldkey group or ASN. A few slices are sourced and sunk by validators, hidden among the rest, so a node can't tell which slices are checked. Calls come about 3 times a day at secret times, backed by a reputation bond. This would replace 1.1's "about 20 full-rate tests a day".

Validators run `bwos-probe` daemons from distributed vantage points using a shared, deterministic probe schedule seeded by the Subtensor block hash (`docs/specs/measurement.md`):

- **Indistinguishable Canary Flows:** Validator canary probes are padded and encapsulated to look identical to `paid` customer traffic on the wire. Transit nodes cannot selectively prioritize probes over customer traffic.
- **Random Saturation Capacity Tests:** Every `CAP_TEST_PERIOD` (`900 blocks`), validators select nodes pseudorandomly for a `CAP_TEST_SECONDS` (`10 s`) saturation burst to measure actual deliverable throughput $B_{\text{meas}}(n)$ against declared capacity $B_{\text{decl}}(n)$:
  - If $B_{\text{meas}}(n) \geq B_{\text{decl}}(n) \cdot (1 - \text{CAP_TOLERANCE_PPM}/10^6)$ (within 10% tolerance), the node receives full credit for its declared capacity: $B_{\text{ver}}(n) = B_{\text{decl}}(n)$.
  - If a node over-claims its physical capacity ($B_{\text{meas}}(n) < 0.90 \cdot B_{\text{decl}}(n)$), it suffers an immediate `OVERCLAIM_PENALTY_PPM` ($200,000$ ppm $= 20\%$) haircut below its measured throughput:
    $$B_{\text{ver}}(n) = B_{\text{meas}}(n) \cdot \frac{10^6 - \text{OVERCLAIM_PENALTY_PPM}}{10^6}$$
    This penalty persists for `CAP_PENALTY_WINDOWS` (`24 windows`), and capacity declarations are subject to a `CAP_CHANGE_WINDOWS` (`24 windows`) cooldown.

---

## 4. Dual-Mechanism Incentive Design

All scoring in Transit Fabric (`crates/bwos-core/src/scoring/mod.rs`) is implemented in **100% deterministic integer fixed-point arithmetic** (`u128`/`i128` parts-per-million, $10^6$ ppm = 1.0, with floor division and deterministic `BTreeMap` UID ordering). Given identical committed inputs, every honest validator on any CPU architecture computes **byte-identical Subtensor weight vectors**.

### 4.1 Mechanism 0 (`mechid=0`): Physical Transit Capacity & Verified Usage

For each transit operator UID $u$ in window $w$:

1. **Verified Fleet Capacity ($K_u$) and Uptime-Weighted Capacity ($\text{cap}_u$):**
$$K_u = \sum_{n \in \text{Fleet}(u)} B_{\text{ver}}(n), \qquad \text{cap}_u = \left\lfloor \frac{K_u \cdot A_u^k}{(10^6)^k} \right\rfloor$$
where $A_u \in [0, 10^6]$ is the operator's probe availability (uptime in ppm) and $k = \text{UPTIME_EXP} = 2$. Squaring uptime strongly penalizes flaky nodes (50% uptime yields 25% capacity credit).

2. **Node Delivery Quality ($Q_n$):**
$$\text{lat_ppm}_n = \min\left(10^6, \left\lfloor \frac{\text{predicted_us}_n \cdot 10^6}{\text{measured_us}_n} \right\rfloor\right)$$
$$\text{loss_ppm}_n = 10^6 - \min\left(10^6, \text{loss_measured_ppm}_n \cdot \text{LOSS_MULT}\right)$$
$$Q_n = \left\lfloor \frac{\text{lat_ppm}_n \cdot \text{loss_ppm}_n}{10^6} \right\rfloor$$
Crucially, $\text{predicted_us}_n$ is computed deterministically by validators from the window's topology snapshot link latencies (`routing::route_predicted_us`), never trusted from an unverified solver estimate. With LOSS_MULT = 10, a node with $\geq 10\%$ packet loss receives $Q_n = 0$.

> **Updated in version 1.1 (ADR-0053):** the usage term $\text{use}_u$ becomes **verified canary carriage only**: validator-allocated, delivered, digest-passing bytes. Paid and free customer bytes never enter it. Until PB-38 lands, Mechanism 0 pays on capacity alone ($W = 10^6$), and the paid-traffic trigger described in 1.0 is retired (PB-40). Steps 3 and 5 below describe the 1.0 design.

> **Status in version 1.2:**
>
> - **Live on testnet 611:** Mechanism 0 pays on uptime-weighted verified capacity alone ($W = 10^6$), and only to nodes with a data plane: a BGP session with a node of another UID (§2.3). With no proven capacity in a window, validators set no Mechanism 0 weights.
> - **Proposed, gated off: locational scarcity pricing (ADR-0057).** Mechanism 0 today pays the same per Mbps wherever the Mbps is, so cheap capacity piles up in a few cities. ADR-0057 multiplies a node's credit by a regional price $G_n$: demand share over supply share, where demand comes from inputs miners can't move. A node's region comes only from validator RTT probes, using the light cone: added delay can make a node look farther from a site, never closer, and regions are separated so that "farther" can never mean "somewhere else". Total emissions stay fixed. Until it turns on, $G_n = 10^6$ for every node.
> - **Proposed, gated off: proof of measurement work (ADR-0055).** A capped slice of Mechanism 0 would go to each validator's measurer UID, for canary and capacity-test bytes it sent that nodes signed for, capped by its stake share. A validator that copies others' revealed results has no such evidence, so measuring pays better than copying.

3. **Quality-Adjusted Verified Usage ($\text{use}_u$):**
$$\text{use}_u = \sum_{n \in \text{Fleet}(u)} \sum_{c \in \{\text{paid}, \text{free}, \text{canary}\}} \left\lfloor \frac{\text{bytes}_{\text{ver}}(n, c) \cdot W_c \cdot Q_n \cdot C_n}{10^{18}} \right\rfloor$$

where class weights are $W_{\text{paid}} = 1,000,000$ ppm, $W_{\text{free}} = 250,000$ ppm, and $W_{\text{canary}} = 100,000$ ppm, and $C_n \in [10^6, C_{\max}]$ is the geographic/ASN diversity multiplier.

4. **Usage Concentration Cap (`USAGE_SHARE_CAP`):** To prevent a single large operator from monopolizing usage emissions, no UID's usage share $\text{use}_u / \sum \text{use}$ may exceed `USAGE_SHARE_CAP` ($250,000$ ppm $= 25\%$) whenever at least $\lceil 10^6/\text{USAGE_SHARE_CAP} \rceil = 4$ UIDs have positive usage. Excess share is iteratively clipped and redistributed pro rata across uncapped UIDs.

5. **Blended Mechanism 0 Share ($\text{share}_u$):**
$$\text{share}_u = \left\lfloor \frac{\text{ALPHA} \cdot \text{cap}_u}{\sum_j \text{cap}_j} \right\rfloor + \left\lfloor \frac{(10^6 - \text{ALPHA}) \cdot \widehat{\text{use}}_u}{\sum_j \widehat{\text{use}}_j} \right\rfloor$$
In 1.0, ALPHA $= 700,000$ ppm (70% bonded capacity, 30% verified traffic usage). Since 1.1 the capacity weight is `W`, which is $10^6$ (capacity only) until PB-38's canary-carriage term lands, and the step-down tied to paid traffic is retired (ADR-0053, PB-40).

---

### 4.2 Mechanism 1 (`mechid=1`): Competitive Multi-Commodity Route Solving

In Mechanism 1, route solvers compete every window via commit-reveal (`hash("route-table", table)`) to optimize the global route table across five distinct QoS traffic classes (`ADR-0017`, `ADR-0018`). In the planned design they also compete on the **peering plan**, which decides which links the fabric runs (§4.2.1).

**Live on testnet 611:** a solver runs `bwos-solver daemon` around its own solver program, in any language. Each window it takes the validators' attested snapshot, solves, signs and commits a manifest naming its table before the deadline, and serves the table at its hotkey's axon for validators to pull. A table must beat the open-source reference solver by `EPSILON_ROUTE` (2%) to be paid.

| Traffic Class | Target Use Case | Composite Objective Weights | Hard SLA Gate |
|---|---|---|---|
| `realtime` | VoIP, trading, gaming, interactive SSH | 70% $G_{\text{lat}}$ + 30% $G_{\text{jit}}$ | RTT≤direct , jitter ≤ 5 ms, loss ≤ 0.5% |
| `low_latency_hd` | 4K video conferencing, cloud desktop | 50% $G_{\text{lat}}$ + 50% $G_{\text{bw}}$ | RTT≤direct , loss ≤ 5.0% |
| `bulk` | AI checkpoint sync, dataset replication | 80% $G_{\text{bw}}$ + 20% $G_{\text{lat}}$ | RTT ≤ 1.50 × direct, loss ≤ 5.0% |
| `economy` | Background backups, low-cost transit | 70% $G_{\text{cost}}$ + 30% $G_{\text{lat}}$ | RTT ≤ 1.50 × direct, loss ≤ 5.0% |
| `resilient` | Mission-critical dual-path HA links | 60% $G_{\text{lat}}$ + 40% ($\mu_{\text{disj}} \cdot G_{\text{backup}}$) | Valid disjoint `backup_path`, loss ≤ 5.0% |

For each sampled demand pair $p = (\text{src}, \text{dst}, \text{class})$, validators evaluate the normalized improvement components clamped to $[-10^6, +10^6]$ ppm relative to the measured direct internet baseline:

- **Latency Gain ($G_{\text{lat}}$):** $(\text{direct_us} - \text{measured_us}) \cdot 10^6 / \text{direct_us}$
- **Jitter Gain ($G_{\text{jit}}$):** $(\text{direct_jitter_us} - \text{measured_jitter_us}) \cdot 10^6 / \text{direct_jitter_us}$
- **Loss-Adjusted Bottleneck Bandwidth Gain ($G_{\text{bw}}$):** $(\text{eff_bottleneck_mbps} - \text{direct_mbps}) \cdot 10^6 / \text{direct_mbps}$
- **Congestion-Adjusted Cost Gain ($G_{\text{cost}}$):** Each link $e$ with utilization $u_e \in [0, 10^6]$ ppm incurs a quadratic congestion surcharge:
$$\text{eff_cost_ppm}(e) = \text{cost_ppm}(e) \cdot \left(10^6 + \left\lfloor \frac{\text{CONGESTION_SURCHARGE_PPM} \cdot u_e^2}{10^{12}} \right\rfloor\right) / 10^6$$
encouraging solvers to spill traffic smoothly across parallel paths before any single bottleneck saturates.
- **Disjoint Backup Multiplier ($\mu_{\text{disj}}$):** For `resilient` routes, $\mu_{\text{disj}} = 1,000,000$ ppm ($1.0\times$) if `backup_path` shares **neither** interior nodes nor directed links with `path`, and $500,000$ ppm ($0.5\times$) if it is link-disjoint but shares an interior node.

#### Solver Score & Anti-Copycat Tournament Selection

A solver $s$'s overall window score is computed using the **lower median** of pairwise gains minus feasibility and route-flapping penalties:

$$\text{score}_s = \text{lower_median}_p\Big(\text{gain}_p(s)\Big) - \text{FEASIBILITY_PENALTY} \cdot \text{overloaded_nodes}(s) - \left\lfloor \frac{\text{CHURN_PENALTY} \cdot \text{churn_ppm}(s)}{10^6} \right\rfloor$$

- **Copycat Exclusion (updated in 1.1):** only solvers that beat the reference solver by more than `EPSILON_ROUTE` earn a rung, and everything at or below the reference is unearned. A copy of last window's production table scores 0. In the 1.0 rule, any non-reference solver whose score is within `EPSILON_ROUTE` ($20,000$ ppm $= 2\%$) of the open-source reference solver (`bwos-solver`), or whose pairwise gain vector duplicates the reference solver or an earlier-committed solver, is eliminated.
- **Winner & Unearned Allocation:** The highest-scoring solver replaces the incumbent only if $\text{score}_{\text{candidate}} \geq \text{score}_{\text{incumbent}} + \text{EPSILON_ROUTE}$. Every table is graded against the next window's snapshot, and solvers are paid on a graded ladder: the top `LADDER_N` (8) positive scores take rungs whose weights halve (`LADDER_DECAY_PPM` $= 500,000$), and a table within `CLONE_DISTANCE_PPM` (2%) of one committed in the same block or earlier scores 0 (`scoring.md` §Mechanism 1, ADR-0035). Any share won by the reference solver is unearned weight: it goes to the owner's disclosed **first-party UID** under a separate coldkey, never to an owner-coldkey hotkey or a burn UID, and is reported publicly (`scoring.md` §Unearned weight and the first-party UID). Unearned **transit** weight (Mechanism 0) goes to no one: with no proven capacity in a window, validators set no Mechanism 0 weights.

##### 4.2.1 Peer discovery: solvers compete on the peering plan (planned)

> **Status:** planned. The design was approved on 2026-11-12 (System One autonomous peering design). It needs two ADRs (peering records and dynamic peers; peering plans in Mechanism 1) and an incentive red team before it is built, and it comes before the bug bounty. Nothing here is built or scored today; peering is self-service (§2.3).

A route table can only use links that exist. Today those links come from operators asking for peers, which is the opposite of what a fabric needs: a topology that is good for the whole network, which no node's own preference decides. The design gives that job to the solvers, who already see the whole snapshot and compete on routing.

- **Solvers compete on two outputs.** Alongside its route table, a solver produces a **peering plan**: the adjacency set, which pairs of nodes peer, for the next window. Whether plan and table are one submission or two, and how a plan pays, is for the ADR to settle.
- **Validators score plans and attest the winner.** Validators grade a plan as they grade tables: on the next window's snapshot, by the network-wide gain it delivers under the tables that ran on it, less a churn penalty and penalties for breaking the diversity rules, the exploration share or per-node caps. The winning plan is attested by validator quorum, so its authority comes from validator consensus, not from any one solver. Nodes apply only a plan with quorum; without one, a node keeps its current peers and its draw.
- **The draw explores; plans exploit.** The block-hash-seeded draw (§2.3) creates and measures new links, and every plan must keep a share of each node's links from its draw. Plans then choose the best of what has been measured, and the topology can't harden around one group.
- **Position hints.** Nodes' and solvers' RTTs add vantage points beyond the validators' sites. They count only as **hints that can only narrow** a validator placement (ADR-0057) or flag a conflict. They never place a node by themselves, because nodes and solvers are paid participants, and m-of-n medians plus the light cone keep any one reporter from moving a node.
- **The self-dealing concern.** A solver could write plans that favor its own operator's nodes. This must be settled by the incentive red team before plans affect anything. The diversity rules, the exploration share and per-node caps are the design's first limits on it; whether they are enough is the red team's question.

- **Rollout.** First, automatic peering by the draw ships with plans in **shadow mode**: solvers submit plans, validators score them and publish a leaderboard, and nodes still follow the draw. Plans pay nothing and aren't a scoring input, which measures how much better solver plans are than the draw. After the red team, plans become binding on testnet and pay through Mechanism 1, activated at a scheduled block (ADR-0049). The bug bounty runs on binding plans, and mainnet gets them only after its findings are fixed.

#### 4.2.2 Sharded solving (proposed)

ADR-0059 (proposed, gated off) splits solving by traffic-class group and destination region, decided by the founder on 2026-11-09. Each shard has its own competition, and the production table is a fixed, published merge of the shard winners that every validator recomputes and attests. It is not a table any company operates. Sharding by destination keeps every route towards a destination inside one shard, so the merged table stays loop-free. With one shard, it reproduces today's Mechanism 1 exactly. Implementation follows the first testnet wave.

---

## 5. Prosumer Edge Fleet Aggregation & The "Mine-to-Surf" Loop (`ADR-0019`) — later phase

> **Later phase, off the launch path.** Launch is transit and route solving only. VPN exits, the client app and fleet-aggregator vouchers come after it (`docs/launch-stages.md`, Stage 5). Vouchers are commercial only and never earn emissions (ADR-0053), and the 1.0 voucher burn is retired. This section keeps the idea in short form.

Bittensor subnets have a fixed number of UID slots (e.g., 256), so desktop and home-lab users can't each register a UID. Two things address that:

- **Today (live):** one UID can bind many nodes: up to 7 inline in its commitment, more through a commitment manifest (§2.1). A home node behind NAT can run as a no-root node or container and be measured and paid like any other.
- **Later phase, Fleet Aggregators:** a device signs a `FleetEnrollment` (domain `fleet-enrollment`) to join an aggregator's UID, which may keep a capped commission (`MAX_AGGREGATOR_COMMISSION_PPM`, 20%) and shares the rest of the UID's emissions in proportion to verified contribution.
- **Later phase, "Mine-to-Surf":** an edge user could turn what their node earns into a `BandwidthVoucher` for VPN exit access, with no card or account. Vouchers here are purely commercial: they authorize traffic at an exit or gateway and never enter scoring.

---

## 6. Tokenomics: Emissions, Wholesale and the Buyer-Side Burn

> **Version 1.3 note (founder, 2026-11-13).** This section is rewritten to the current model. Removed: the 1.1 model's 80/20 company buyback, the company's 20% fee on protocol revenue, the `ALPHA` step-down tied to the buyback, the subsidy ratios, and the three-year revenue, market-cap and company-income projections (1.2's §6.2 and §6.3). Their premises are retired (ADR-0053, ADR-0063, ADR-0064, ADR-0065), and the current model has no projection to replace them. In their place, this section says who pays whom, in what unit, and what is burned.
>
> **Status.** Emissions-only pay is accepted (ADR-0053) and live on testnet 611 as capacity pay (§4.1). The buyer-side burn is the decided direction: the founder approved ADR-0063's direction and chose burn over recycle on 2026-11-11 (OQ-288). ADR-0063 itself is still *proposed* (revision 5), pending its red team and PB-49's simulations. US-dollar quoting (ADR-0065) is accepted in principle, with its details proposed. None of the wholesale path is built (PB-49).

### 6.1 Two flows that never meet (ADR-0053)

Transit Fabric keeps protocol pay and customer money in two separate flows. Nothing from the second ever enters the first as a scoring input.

1. **The emissions flow (live on testnet).** Nodes, gateways, solvers and validators are paid only in the subnet's alpha emissions, allocated by what validators measure themselves: capacity and uptime today, canary carriage once PB-38 scores it (Mechanism 0), and routing quality (Mechanism 1). No node is paid for carrying traffic directly. The economic model's inputs (`docs/economics/README.md`, as of 2026-10-17) are Bittensor's `3,600 TAO/day` of network emissions and the per-subnet split of **41% to miners** (here, nodes and solvers), **41% to validators** and **18% to the subnet owner**. The owner's share is the subnet owner's own; it is not BandwidthOS revenue and not a protocol fund.
2. **The commercial flow (designed, ADR-0063 proposed).** Customers pay resellers. Resellers buy wholesale capacity by burning TAO, and pay gateways a port fee. Payments, vouchers, credits, client receipts and the rate card are **never a scoring input**, so faking, buying or steering "paid" traffic earns nothing, and mishandling customer traffic can only lower a score (ADR-0053 §2).

**How customer money reaches nodes and solvers: through the pool, never through receipts** (ADR-0053 §4, ADR-0063). A wholesale purchase buys the subnet's alpha from its pool with TAO and destroys it. Each purchase puts TAO into the pool and removes the alpha it bought. A subnet's share of network emissions depends on its pool: the economic model takes it as set by the alpha price, and ADR-0065 counts a burn as TAO flow into the pool for the emission share. Either way, more customers make every participant's emissions worth more. Nobody, the company included, can move any participant's slice: allocation within the subnet is set only by measurement (ADR-0049, ADR-0053). One caution from ADR-0063 (OQ-288): `SubnetAlphaOut` doesn't drop on a burn, so a supply figure derived from it overstates circulating alpha.

### 6.2 Who pays whom, in what unit, and what is burned (ADR-0063, ADR-0065)

| Payer | Payee | For | Quoted in | Paid in | Burned? |
|---|---|---|---|---|---|
| End customer | A reseller (BandwidthOS, a VPN, an ISP, a broker) | The reseller's product: transit now, exits in a later phase | The reseller's choice | Any currency the reseller accepts (wire, card, USDC) | No. The protocol never touches fiat or USDC |
| Reseller | Nobody: the subnet pool | The **fabric fee**: one window of a lane's rate cap and byte quota | Micro-US-dollars (`uusd`, $10^{-6}$ USD) per GiB, on the validators' rate card | TAO, by `SubtensorModule.add_stake_burn` from the reseller's own coldkey | **Yes.** The TAO buys alpha from the pool, and that alpha is destroyed in the same call |
| Reseller | Each lane gateway's registered coldkey | The **port fee**, per window, before the next auction | `uusd` per GiB, from the gateway's signed registration, fixed per credit by the card | TAO, by `Balances.transfer_keep_alive` | No. It is the gateway's revenue |
| The chain's emissions | Nodes, gateways, solvers, validators and the owner | Measured work (§4) | — | Alpha, every 360-block tempo | No |

```mermaid
flowchart LR
    Cust["End customers\n(any currency)"]
    Res["Resellers\n(BandwidthOS, VPNs, ISPs, brokers)"]
    Card["Validator-signed rate card\n(lanes, uusd prices, one TAO/USD reference)"]
    Burn["add_stake_burn, signed by the reseller\nTAO buys α from the pool; α destroyed"]
    Pool["Subnet pool: TAO in,\nbought α destroyed"]
    Em["Emissions: nodes, gateways,\nsolvers, validators, owner"]
    GW["Gateways\n(port fee in TAO, per window)"]

    Cust --> Res
    Card --> Res
    Res --> Burn
    Res --> GW
    Burn --> Pool
    Pool --> Em
```

**Three layers, each with one job** (ADR-0063 §1). The **protocol** publishes the rate card, reads purchases from the chain and publishes credits, balances and a burn ledger; nobody operates it and it takes nothing but burned TAO. **Resellers** sell to end customers, buy TAO, make purchases and issue `BandwidthVouchers`. **Gateways** (registered datacenter operators, ADR-0052) on-ramp and off-ramp traffic, meter it at both ends of a lane and enforce credits. Currency conversion is the reseller's job. There is **no treasury, no custodian and no protocol fee**: at the protocol level no one receives revenue except gateways, which receive their own port fees.

**The rate card** (ADR-0063 §2, amended by ADR-0065 §4). Each window, validators publish a signed card. A lane is an ordered pair of gateways run by different operators. v0 sells one class, `best_effort`: a byte quota plus a rate cap. For each lane:

$$\text{headroom} = \min(\text{min_cut}, \text{egress}) \cdot \frac{\texttt{WHOLESALE_SELLABLE_PPM}}{10^6} - \text{sold rate}$$

$$\text{price}_{\text{uusd/GiB}} = \frac{\texttt{WHOLESALE_FLOOR_UUSD_PER_GIB} \cdot \text{curve}(\text{utilization_ppm})}{10^6}$$

where `min_cut` is the min-cut between the two gateways over verified capacity, `egress` is the egress gateway's measured throughput, and `curve` is a published integer piecewise-linear function that rises as the lane fills. **To be set:** the floor and the minimum purchase are calibrated in `uusd` before mainnet, and the curve breakpoints and `WHOLESALE_SELLABLE_PPM` are PB-49 parameters. None has a value yet.

**Clearing** (ADR-0063 §3). Each window's bids clear once, as a uniform-price batch auction per lane, ranked renewals first, then by `max_price`, then by a hash seeded after the cutoff. New fills pay the higher of the curve at final utilization and the highest `max_price` left unfilled for lack of headroom. Renewals pay the higher of the curve and the new-fill price capped at `WHOLESALE_MAX_RENEWAL_STEP_PPM` (proposed 10%) above their previous price. The price-setting unfilled bid forfeits `WHOLESALE_BID_BOND_PPM` (proposed 5%) of its fee. A credit runs one window at a time, with no fixed term.

**A burn is final** (ADR-0063 §3, revision 4). Nothing burned is refunded for unused bytes, a lapse, deregistration or a failed guard. Balance, held in `uusd` and owned by the payer coldkey, comes only from paying above the clearing price, being outbid, rounding remainders and explicit top-ups, and is spent by later bids.

**Conversion: quoted in dollars, settled in TAO** (ADR-0065 §3). The card for window $v$ carries one reference, `tao_usd_ref`: the chain's `TaoUsdPrice.accepted` read at the window's first block, in units of $10^9$ per USD. Every validator reads the same value. A burn of `rao` included in window $v$ is credited, rounded in the network's favour, as

$$\text{uusd} = \left\lfloor \frac{\text{rao} \cdot \texttt{tao_usd_ref}}{10^{12}} \right\rfloor, \qquad \text{and a fee due in uusd costs} \quad \text{rao} = \left\lceil \frac{\text{uusd} \cdot 10^{12}}{\texttt{tao_usd_ref}} \right\rceil.$$

*Worked example, with ADR-0065's own reference figure:* at \$211.00, `tao_usd_ref` = `211000000000`. A burn of 1 TAO ($10^9$ rao) is credited $\lfloor 10^9 \cdot 211{,}000{,}000{,}000 / 10^{12} \rfloor =$ `211,000,000` uusd, which is \$211.00. The credit is fixed at inclusion; if TAO moves afterwards, it doesn't change. Resellers carry TAO/USD risk only between buying TAO and burning it, at most two windows. If the

oracle is stale for more than `WHOLESALE_ORACLE_MAX_AGE_BLOCKS` (300), the card is *held* at the last fresh reference for up to `WHOLESALE_ORACLE_HOLD_WINDOWS` (3) windows, then *unavailable*. While it is held, new purchases and top-ups aren't credited, but renewals and balance spends continue. Once it is unavailable, only balance spends clear, and renewals that need a burn are suspended, not lapsed. Bittensor's Gamma credits are a later option, adopted only by a new ADR.

*A real burn on testnet 611* (ADR-0063, 2026-11-11): a coldkey that is not the subnet owner sent one `batch_all` of `add_stake_burn` and a remark. It put 10,000,000 rao (0.01 TAO) into the pool, bought and burned 77,214,706 alpha, moved spot from 0.129362 to 0.129526 TAO per alpha, and paid a transaction fee of 993,533 rao (about 0.001 TAO, fixed per purchase).

**Guards for alpha holders** (ADR-0063 §3a; values from the OQ-291 replay of 8 young mainnet subnets, `docs/economics/oq-291-guard-calibration.md`). The protocol can't stop anyone's burn; it controls what it credits, and a burn that fails a guard is forfeited, never refunded, so evading a guard never pays. New purchases and top-ups face every guard; renewals face only the short-horizon band, the slippage cap and the size cap, so customers already served can't be cut off by a multi-day trend.

| Guard | Parameter | Value |
|---|---|---|
| Reference price: median spot over about 7 days, sampled at blocks chosen after each window | `WHOLESALE_REF_WINDOWS` / `WHOLESALE_REF_SAMPLES_PER_WINDOW` | 140 windows / 4 (proposed) |
| Price band: a purchase's `limit` at most this far above the reference | `WHOLESALE_PRICE_BAND_PPM` | 25% (from OQ-291; was 15%) |
| Volatility gate: closes new purchases when the p90–p10 spread over the reference exceeds it | `WHOLESALE_MAX_VOL_PPM` | 40% (from OQ-291; was 30%) |
| Short-horizon band: execution price over the 50-block median | `WHOLESALE_SHORT_BAND_PPM` / `WHOLESALE_SHORT_REF_BLOCKS` | 5% / 50 blocks (proposed) |
| Slippage cap: execution over the simulated swap at the parent block | `WHOLESALE_MAX_SLIPPAGE_PPM` | 2% (proposed) |
| Size cap: one extrinsic's burns as a share of the pool's TAO reserve | `WHOLESALE_MAX_BURN_PPM` | 1% (proposed, calibrated on testnet) |
| Notice before any economic parameter, the floor or the curve changes | `WHOLESALE_PARAM_NOTICE_WINDOWS` | 600 windows (about 30 days) |

The OQ-291 replay found that at 25%/40% the guard for new purchases was closed in about 14% of windows over a subnet's first 90 days, and about 7% over days 30 to 90, while still closing 92% of windows that jumped more than 25% above the previous day's median. A subnet's launch month stays about 37% closed under any setting; resellers ride that out on a balance buffer (ADR-0064 §6).

**The burn ledger** (ADR-0063 §3b). Each window, validators publish one row per burn (payer, TAO in, alpha burned, outcome) and two identities anyone can recompute from the chain: TAO burned = credited + balance created + forfeited + held pending, and credited + balance spent = value of windows held. The company's own wholesale coldkeys are listed in it.

**One disclosed asymmetry** (ADR-0063 §1, red team L1). Every reseller pays the same clearing price, but a buyer that holds alpha gets back part of its fee through the price rise its own purchase causes. Every buyback works this way; it is disclosed, not fixed.

### 6.3 What BandwidthOS earns, and what nobody earns (ADR-0063 §6, ADR-0064)

- **The protocol earns nothing.** It holds no money and charges no fee. The fabric fee is burned.
- **Gateways** earn their port fees, paid straight to their registered coldkeys window by window, plus emissions like any node.
- **Nodes, solvers and validators** earn emissions only (§6.1).
- **BandwidthOS earns a reseller's margin, and nothing from the protocol.** It pays the clearing price like every reseller and gets no early access to parameters or data. Protocol revenue never passes through it. For each window, its margin is

$$\text{margin} = \text{retail revenue} - \sum_{\text{credits}} \text{fabric fee} - \sum_{\text{credits}} \text{port fees} - \text{its own costs}$$

where the fabric fee and port fees are set by the card's clearing prices, and its own costs include buying TAO and the transaction fee on each purchase. None of these terms has a published value yet, so the paper gives no figure for it. Under its published treasury policy (ADR-0064 §6, ADR-0063 §6), its alpha comes only from retail margin and sits on hotkeys that never set weights; its holdings are disclosed quarterly; it keeps a balance buffer of a set number of windows; it sells no alpha within `N` windows (proposed 140, about 7 days) of its own wholesale purchases, and at most 5% a month on an announced schedule. Any corporate buyback it chooses to run is kept out of the fabric burn ledger. Nodes BandwidthOS may run are a reference deployment and are paid like anyone's (§7.2); how that pay sits with the treasury rule is open (OQ-301).

**What the old projections covered and the current model doesn't.** The 1.1 model's VPN backbone and pay-per-GB exit revenue depend on exits, a later phase. The telemetry feed (§7.1) is a later phase: it has no settlement path in ADR-0063. The repository's historical economic model still assumes the retired buyback and company fee, so this section uses only its emission inputs.

---

## 7. BandwidthOS, the Model Home: Commercial Operations & Go-To-Market Workflows

> **Version 1.2 note (ADR-0062, ADR-0064).** Transit Fabric is the open protocol and network; it is not a business. **BandwidthOS** is the one company the founder runs around it: the first reseller and first gateway, run in public as a working example anyone can copy. Everything it runs to operate (sign-up and accounts, billing adapter, TAO procurement, wholesale buying, voucher issuance, the customer dashboard, gateway operation and runbooks) is published as an open-source reference reseller under Apache-2.0. It has **no protocol privilege**: it pays the clearing price like every reseller, gets no early access to parameters or data, and gets no peering advantage. It competes on operations, support and brand. It earns a reseller's margin, never protocol revenue (§6.3). Whether the company is an Inc. or an LLC is not yet decided. Nothing in this section is built yet: wholesale is PB-49, the reference reseller PB-50, gateways PB-39.

### 7.1 Market Positioning: Programmable NaaS & Middle-Mile SD-WAN

While the Transit Fabric Bittensor subnet provides the decentralized incentive, measurement, and settlement layer, enterprise buyers, AI cloud operators, and commercial VPN providers need a standard commercial counterparty. Under ADR-0063 any reseller can be that counterparty on the same terms; **BandwidthOS** is the first. It positions itself as **Programmable Network-as-a-Service (NaaS) and Middle-Mile SD-WAN**, at the market intersection of **Megaport / PacketFabric** (elastic inter-datacenter transit) and **Cloudflare Magic WAN / Argo** (real-time latency/jitter-optimized routing), backed by the fabric's open supply.

A reseller can sell two core products and, later, two by-products:

1. **Aggregated Transit on Demand (Elastic Middle-Mile), planned:** Customers establish a single eBGP/IPsec/WireGuard session into a registered gateway (ADR-0052), or reach it over a cross-connect or proxy, and gain burst transit across the global fleet of Mechanism 0 operators. The aim is pricing 30–50% below hyperscale egress and legacy SD-WAN contracts, because underlying miners are monetizing sunk 95th-percentile commit headroom. The zero-touch 1U `Transit Fabric Edge` appliance is a later phase.
2. **Near-Real-Time QoS-Based Routing ("BGP with a Brain"), planned as a product:** Powered by Mechanism 1 route solvers and validator measurement, customers steer traffic across five SLA-enforced QoS classes (`realtime`, `low_latency_hd`, `bulk`, `economy`, `resilient`) with sub-second BFD failover onto interior-node-disjoint backup paths (520–680 ms measured over the internet in Phase A).
3. **Turnkey Multi-Region VPN Exit & Edge Access, later phase:** Wholesale and retail exit access via registered Exit Partners (`ADR-0004`) authorized by `BandwidthVouchers`, which are commercial only.

4. **Global Link-Metric Telemetry Feed, later phase:** pairwise latency, jitter, packet loss, and link-saturation matrices from validator measurement, for trading desks, CDNs, and WAN observability platforms.

### 7.2 BandwidthOS Nodes: a Reference Deployment (planned)

> **Version 1.3 note (founder, 2026-11-13; ADR-0064 §8).** This replaces 1.1's company-owned "Anchor Core Routers" and 1.2's hub-node text, including its projection of capacity pay against hub transit cost, which rested on the retired `ALPHA`.

BandwidthOS may run nodes, but only as a **reference implementation**: open-source code and a public example that others copy, so they can run nodes and resell as well. Its nodes:

- **Run the same open-source stack as anyone's** (for a kernel node: `bwos-node`, FRRouting `bgpd`/`bfdd`, WireGuard and the `bwos-node tune-host FQ/BBR` profile), and how they are deployed is published with the rest of the reference implementation (ADR-0064 §3).
- **Get no protocol privilege** (ADR-0064 §4). They are measured, paid and peered under the same Mechanism 0 rules as every node: self-service peering today, the draw and solver peering plans later (§2.3, §4.2.1). They hold no special role: not an anchor, an ingress point, a route reflector or a peering hub. The fabric is flat eBGP with no route reflectors, and no node or company chooses the topology. A BandwidthOS node is a gateway only if it registers as one under ADR-0052's published criteria, which any datacenter operator can meet.
- **Earn emissions like any operator's, and buy like any reseller.** What its nodes earn is ordinary operator pay, and when BandwidthOS resells capacity it burns TAO at the same clearing price as everyone else (ADR-0063). Its node UIDs have no special standing and are not separately disclosed (ADR-0064 §8).

Today on testnet, the team runs public nodes that peer with testers' nodes on request (§2.3), and every validator on subnet 611 is the team's own until outside validators join.

### 7.3 Transaction Clearing & Settlement Workflow (Fiat/USDC to a TAO Burn)

> **Version 1.2 note (ADR-0063, ADR-0065):** this replaces the 1.1 workflow, in which the company bought alpha with customer revenue. Proposed details, pending a red team; not built (PB-49, PB-39).

Enterprise customers never need to custody TAO or $\alpha$ or manage Subtensor hotkeys. A reseller such as BandwidthOS clears transactions end-to-end:

1. **Commercial Invoicing:** The customer pays the reseller in any currency it accepts (wire, ACH, card or USDC) under a monthly commit or pay-per-GB agreement.
2. **Wholesale Purchase:** The reseller bids for a lane on the validators' rate card, priced in US dollars. Its own coldkey signs one `batch_all`: `add_stake_burn`, burning TAO converted at the window's TAO/USD reference, plus a remark naming the credit. The window clears as a batch auction, and anything paid above the clearing price comes back as a dollar balance.
3. **Credit and Quota Issuance (`BandwidthVoucher`):** Validators read the purchase from the chain and publish the credit. The reseller issues `BandwidthVouchers` (`domain: "bandwidth-voucher"`) scoped to its customer's keys, which the lane's gateways meter and enforce.
4. **Per-Window Renewal and Port Fees:** A credit runs one window at a time, renewed by further bids, and each window's port fee is paid in TAO straight to each lane gateway's registered coldkey.
5. **Miner Payouts Through Emissions:** Validators never pay miners from receipts or vouchers: nodes and solvers are paid in $\alpha$ emissions from measured capacity, availability and, once scored, canary carriage, every 360-block tempo. Customer money reaches them through the burn's effect on the subnet's pool, with no accounts-payable process across hundreds of independent operators.
6. **The Reseller's Margin:** What the reseller keeps is its customer revenue less the fabric fees it burned, the port fees it paid and its own costs (§6.3).

## 7.4 Go-To-Market Sequences & End-to-End User Stories

> **Version 1.3 note.** These sequences are illustrative, with fictional names, and were first written for 1.1. Their contract figures stand. Where a step relied on company-owned anchor routers or on the retired buyback, it follows ADR-0063 and ADR-0064, and 1.3 removes the 1.1 figures that rested on the buyback or on `ALPHA` (§6). Sequence 3's appliance and all of Sequence 4 are later phases.

#### Sequence 1 (Beachhead 1 — Months 1–6): Reference Nodes & Onboarding VPN Providers

- **Actors: BandwidthOS** (the first reseller and gateway), **Lantern VPN** (a 40-city crypto-accepting commercial VPN provider), and **Ridgeline Hosting** (a regional colocation operator with 6 Gbps of idle 95th-percentile transit headroom).
- **End-to-End Workflow:**
    1. **Reference Nodes:** BandwidthOS brings online 6 nodes (`100 Gbps` aggregate commit) in Ashburn, Chicago, Frankfurt, Amsterdam, Dallas, and Tokyo as a reference deployment (§7.2): open source, published for others to copy, and measured and paid like anyone's.
    2. **Pilot Onboarding:** BandwidthOS onboards **Lantern VPN** onto a 30-day pilot. Lantern runs `bwos-node` on 10 of its busiest ingress/exit servers, which peer into the fabric by the same process as every node.
    3. **Cryptographic SLA Proof (`PartnerPilotReport`):** Over 4 weeks, validators probe every route and BandwidthOS generates an automated weekly `PartnerPilotReport` (`crates/bwos-core/src/validator.rs`) comparing Lantern's middle-mile traffic over Transit Fabric (`realtime` and `low_latency_hd` classes) against direct public BGP—demonstrating a `20–35%` reduction in transatlantic p95 RTT and jitter and sub-300 ms BFD failover during carrier brownouts.
    4. **Dual-Role Conversion (Customer + Exit Partner):** Lantern signs a `$10,000/month` commercial backbone contract with BandwidthOS (paid in USDC/USD) to carry inter-city VPN traffic over Transit Fabric, **and**, in a later phase, registers its 40 regional servers as Transit Fabric **Exit Partners** (`ADR-0004`).
    5. **Demand Reaches the Subnet:** BandwidthOS buys the wholesale lanes Lantern's traffic needs by burning TAO at the rate card's clearing price (§7.3). How much of Lantern's `$10,000/month` is burned depends on the card's clearing prices and port fees, which aren't set yet (§6.2); the rest is BandwidthOS's margin and costs (§6.3). Seeing the subnet's emissions and demand, **Ridgeline Hosting** registers a Mechanism 0 UID, peers its 3 Gbps idle commit into the fabric, and earns capacity emissions. Lantern's paid traffic itself earns no one anything (ADR-0053).

#### Sequence 2 (Beachhead 2 — Months 3–9): Distributed AI / Neo-Cloud GPU Cluster Interconnects

- **Actors: TensorForge Cloud** (a distributed GPU provider operating H100/B200 clusters across Illinois, North Carolina, and Eastern Europe) and **Mechanism 1 Route Solvers**.
- **End-to-End Workflow:**
    1. **Problem:** TensorForge must synchronize 250 GB model checkpoints every hour between regional datacenters and stream low-latency token inference across nodes, but public BGP suffers from single-path TCP congestion and hyperscale cloud egress fees are prohibitive.
    2. **Peering & QoS Tagging:** TensorForge deploys `bwos-node` on its border gateways (`bwos-node tune-host -if eth0` enabling 64 MB socket buffers, `fq` pacing, and `BBR`), reaches the fabric through a registered gateway (ADR-0052), and tags checkpoint replication as `class = "bulk"` (80% bottleneck bandwidth / 20% latency) and inference streams as `class = "realtime"` ($\leq$ 5 ms jitter).
    3. **Capacity-Aware Solver Execution:** Mechanism 1 solvers (`routing::solve_capacity_aware`) compute multi-commodity flow tables that strip out congested links using quadratic congestion pricing (`CONGESTION_SURCHARGE_PPM`) and stripe TensorForge's bulk transfers across parallel Mechanism 0 transit paths.
    4. **Clearing:** TensorForge pays BandwidthOS a monthly commit + per-TB burst fee in USD; BandwidthOS buys the lanes by burning TAO and issues `BandwidthVouchers` (§7.3).

#### Sequence 3 (Beachhead 3 — Months 6–18): Enterprise HA WAN (`resilient`) & Telemetry Data Buyers

- **Actors: Coastline Labs** (an enterprise linking three offices and a private compute cage) and **QuantPulse** (a low-latency financial infrastructure desk).
- **End-to-End Workflow:**
  1. **Zero-Touch Appliance Deployment (later phase):** Coastline racks a pre-flashed 1U `Transit Fabric Edge` appliance (`docs/adr/0009-edge-appliance.md`) at each site and enters the 1-line `bwos-node claim-code` token into BandwidthOS's portal. The appliance generates its `fc00::/8` identity locally; Coastline's IT team signs the `EdgeConfig` off-box (`domain: "edge-config"`), and no wallet keys ever touch the router.
  2. **Interior-Node-Disjoint HA (`resilient`):** Coastline's inter-site traffic runs under `class = "resilient"`, which requires Mechanism 1 solvers to provision both a low-latency primary path and an **interior-node-disjoint** `backup_path` (`BACKUP_INTERIOR_DISJOINT_MULT_PPM = 1,000,000`). When an upstream carrier link degrades, BFD (`3 × 200 ms`) fails traffic over to the disjoint backup path in under a second (measured 520-680 ms over the internet in Phase A; zero dropped TCP sessions is a goal, not measured).
  3. **Telemetry Monetization (later phase): QuantPulse** subscribes to a telemetry API (`$4,000/month`), consuming the live, validator-signed matrix of global inter-datacenter latency, jitter, and packet loss across all probed ASNs.

#### Sequence 4 (later phase): Prosumer Desktop, OpenWrt & "Mine-to-Surf"

- **Actors: Maya** (a remote systems engineer on macOS and an OpenWrt home router) and a **Fleet Aggregator UID**.
- **End-to-End Workflow (later phase, §5):**
  1. **Onboarding:** Maya installs the Transit Fabric desktop client on her Mac (`bwos-client app`). Her client derives its `/128` `fc00::/8` address and connects to the fabric through an ingress node.
  2. **Mine-to-Surf Enrollment (`ADR-0019`):** Maya signs a `FleetEnrollment` (`commission_ppm <= 20%`). While her Mac or OpenWrt router is online, it carries light transit and earns a share of the aggregator's Mechanism 0 emissions.
  3. **Exit Access:** Her earnings, or exit credits she buys, back a `BandwidthVoucher` that authorizes traffic at Lantern VPN's exit nodes. The voucher is commercial only and carries no protocol burn. In the client, Maya picks an exit region and routes her Mac's TLS 1.3 and SOCKS5 streams across the `fc00::/8` fabric to it.

---

## 8. Security, Privacy & Wire-Obscurity Analysis

### 8.1 Resistance to Economic & Protocol Attacks

| Threat Vector | Protocol Defense Mechanism | Economic / Verification Outcome |
|---|---|---|
| **Self-Dealing / Wash-Trading** | Since 1.1, customer traffic is never credited (ADR-0053): paid bytes earn no emissions, so faking or steering them earns nothing. Canary carriage is validator-allocated (ADR-0051) | Zero gain by construction (1.0 measured `0.96x` under the retired voucher-burn design) |
| **Hop Counter Inflation** | Bilateral `HopReceipt` co-signing (`0.5%` tolerance) + multi-hop chain conservation audit | Inflating interior hop is deterministically zeroed |
| **Missing Middle Receipt** | Contiguous path receipt verification $(n_0, n_1)...(n_{k-1}, n_k)$ | Entire incomplete path segment is zeroed |
| **Capacity Over-Claiming** | Random 10s saturation capacity tests (`CAP_TEST_PERIOD = 900`) + `OVERCLAIM_PENALTY_PPM` (`20%` haircut for 24 windows) | Over-claiming yields `0.73x` return per real Mbps vs. `1.00x` honest |
| **Selective Canary Prioritization** | Goal, not yet true: canaries still differ from paid flows in source, timing and size (threat model T19, Phase B Task 14); $Q_n$ penalizes latency/loss inflation | Dropping customer traffic degrades $Q_n$ and zeroes usage credit |
| **Solver Copycatting / Sybil** | Commit-reveal + `EPSILON_ROUTE` (`2%`) margin + identical `gains_ppm` vector deduplication | Copycat solvers receive `0` Mechanism 1 weight |
| **BGP Prefix Hijacking** | `/128` `fc00::/8` IPv6 address is `0xfc \|\| SHA-512(SHA-512(pubkey))[0..15]` | Cryptographically impossible without Ed25519 private key |

| Threat Vector | Protocol Defense Mechanism | Economic / Verification Outcome |
|---|---|---|
| **Exit Node SSRF / Local Egress** | Strict target IP validation ( `is_forbidden_egress_ip` blocks RFC 1918, loopback, link-local `169.254.0.0/16`, `fc00::/8`, and IPv4-mapped IPv6 `::ffff:0:0/96`) | Returns `HTTP 403 Forbidden` before socket connect |

### 8.2 Wire-Level Privacy ("Entry ≠ Exit") — later phase

> **Version 1.2 note:** these properties concern client traffic through exits, a later phase. Nothing on testnet 611 leaves the overlay. Item 2 is corrected per ADR-0040: Transit Fabric makes no claim that node links resist DPI or blocking.

Compared to traditional single-hop VPNs (where a single server sees both the user's residential IP and their destination traffic, and whose IP address is publicly cataloged on commercial VPN blocklists), Transit Fabric's flat eBGP architecture would provide these structural privacy properties:

1. **Compartmentalized Metadata:** The **ingress node** knows the client's outer IP address but only sees packets destined for an `fc00::/8` exit address. **Interior Transit Nodes** see only encrypted `fc00::/8 -> fc00::/8` backbone frames. The **Exit Node** connects to the public destination ( `:443` ), but sees only the client's `fc00::/8` overlay address—never the client's physical IP—while the payload remains end-to-end TLS 1.3 encrypted from the client's OS.
2. **Not on exit blocklists (corrected in 1.2):** A Mechanism 0 transit node never makes outbound connections to websites, so its address doesn't appear on commercial VPN *exit* blocklists. That is all it gives. The entry connection is WireGuard today, which DPI middleboxes can identify and block, so Transit Fabric claims no censorship resistance for it (ADR-0040).
3. **Wire fingerprint (corrected 2026-11-01):** datacenter backbone links that run IPsec ESP ( `RFC 4303` ) or MACsec look like enterprise interconnects. WireGuard links, which every home and no-root node uses, are easy to identify and don't share that property (ADR-0040).

---

## 9. Empirical Evaluation & Multi-Region Benchmarks

### 9.1 50-Window Adversarial Incentive Simulation ( `bwos-sim` )

We evaluated Mechanism 0 and Mechanism 1 over 50 consecutive windows ( `make sim` ) across 9 competing operator archetypes. Measuring **score share per unit of real physical Mbps supplied** (normalized to Honest = `1.00x` ), every adversarial strategy underperforms honest participation:

| Operator Strategy | Real Capacity Supplied | 50-Window Mean Score Share | Efficiency vs. Honest ( `1.00x` ) |
|---|---|---|---|
| **Honest Operator** | 1,000 Mbps | 20.69% | `1.00x` **(Baseline)** |
| **Self-Dealer (Wash-Trader)** | 1,000 Mbps | 19.77% | `0.96x` *(1.0 design, penalized by voucher burn & cap; since 1.1 paid traffic earns nothing)* |
| **Receipt Inflator** | 1,000 Mbps | 14.23% | `0.69x` *(inflated hop chains zeroed)* |
| **Over-Claimer (3× fake claim)** | 1,000 Mbps | 15.14% | `0.73x` *(* `OVERCLAIM_PENALTY_PPM` *haircut)* |
| **Flaky Operator (60% uptime)** | 1,000 Mbps | 11.49% | `0.56x` *(quadratic* `UPTIME_EXP = 2` *penalty)* |
| **Canary-Gaming Operator** | 1,000 Mbps | 14.23% | `0.69x` *($Q_n$ latency/loss gate)* |
| **Sybil Operator (split across UIDs)** | 1,000 Mbps | 20.52% | `0.99x` *(linear invariance; extra registration cost)* |

### 9.2 Live Multi-Region Hardware-in-the-Loop (HIL) Benchmarks

> **Version 1.2 note:** these runs used the 1.0 prototype client and exit path. Client access and exits are a later phase, so Table 2 measures the overlay prototype, not a service available today.

We deployed `bwos-node` across three live cloud datacenter regions—**US-Central (`US-IL-1`, Chicago, Illinois)**, **US-East (`US-NC-1`, Raleigh, North Carolina)**, and **Europe (`EU-RO-1`, Cluj-Napoca, Romania)**—and connected a live macOS desktop client (`bwos-client app`, `fce0:7248:...`) over Wi-Fi (`en1`) using Layer-4 HTTPS `CONNECT` (`127.0.0.1:7781`) and `SOCKS5h` (`127.0.0.1:7782`) system routing.

**Table 1: In-Datacenter `bwos-node` Engine Overhead (`US-IL-1`)**

| Metric | Measured Value | Notes |
|---|---|---|
| **Control-Plane + Receipt Verification RTT ($p_{50}$)** | `0.164 ms` (`147 µs` min) | 50 consecutive queries on `US-IL-1` (`Linux 6.8.0-79-generic`) |
| **Control-Plane + Receipt Verification RTT ($p_{99}$)** | `1.157 ms` | Includes JSON canonicalization & Ed25519 verification |
| **Node Direct Egress Line Rate (50 MB TLS 1.3)** | `766.8 Mbps` (`0.522 s`) | Sustained TLS 1.3 stream to Cloudflare CDN |

**Table 2: End-to-End Multi-Region Overlay Performance (`macOS Client ⟶ Ingress ⟶ BGP Transit ⟶ Exit ⟶ Internet`)**

| Selected Regional Exit | Overlay Hops | Verified Public Egress IP & Cloudflare PoP | Signed `ClientReceipt` + Co-Signed `HopReceipt` RTT ($p_{50}$) | End-to-End TLS 1.3 Handshake (`:443`) | Sustained 10 MB TLS 1.3 Stream (to macOS Wi-Fi) |
|---|---|---|---|---|---|
| **US-Central (`US-IL-1` · Illinois)** | **2 hops** | `(address redacted)` (`US` · PoP: `MSP`) | **314.9 ms** (`313.3 ms` min) | **446.7 ms** | **53.90 Mbps** (`1.48 s`) |
| **Europe (`EU-RO-1` · Romania)** | **4 hops** | `(address redacted)` (`RO` · PoP: `FRA`) | **759.6 ms** (`755.4 ms` min) | **824.5 ms** | **35.72 Mbps** (`2.24 s`) |
| **US-East (`US-NC-1` · North Carolina)** | **3 hops** | `(address redacted)` (`US` · PoP: `IAD`) | **539.2 ms** (`441.7 ms` min) | **569.2 ms** | **27.69 Mbps** (`2.89 s`) |

---

## 10. Normative Protocol Parameters

| Parameter | Default Value | Normative Meaning |
|---|---|---|
| `ALPHA` | `700_000` ppm (70%) | 1.0 value. Replaced by `W`, which is `1_000_000` (capacity only) until PB-38's canary-carriage term lands; the paid-traffic step-down is retired (ADR-0053, PB-40) |
| `UPTIME_EXP` | `2` | Exponent $k$ on probe availability $A_u$ in capacity scoring |
| `W_PAID` / `W_FREE` / `W_CANARY` | `1_000_000` / `250_000` / `100_000` ppm | 1.0 traffic class weights. Paid and free bytes are never credited since 1.1 (ADR-0053); the canary-carriage term is set by PB-38 |
| `USAGE_SHARE_CAP` | `250_000` ppm (25%) | Maximum usage share any single UID may capture |
| `RECEIPT_TOLERANCE_PPM` | `5_000` ppm (0.5%) | Maximum allowed counter divergence between adjacent hops |
| `CAP_TOLERANCE_PPM` | `100_000` ppm (10%) | Allowed shortfall in random saturation capacity tests |

| Parameter | Default Value | Normative Meaning |
|---|---|---|
| `OVERCLAIM_PENALTY_PPM` | `200_000` ppm (20%) | Haircut below measured capacity when a node over-claims |
| `EPSILON_ROUTE` | `20_000` ppm (2%) | Required improvement margin to unseat the incumbent route table |
| `LADDER_N` / `LADDER_DECAY_PPM` | `8` / `500_000` ppm | Mechanism 1 ladder: rungs and each rung's weight relative to the one above (replaces `WINNER_SHARE`) |
| `CLONE_DISTANCE_PPM` | `20_000` ppm (2%) | A table this close to an earlier-committed one scores 0 |
| `OVERLAY_BASE_ASN` | `4_200_000_000` | Base RFC 6996 4-byte private ASN for `fc00::/8` eBGP peers |
| `MAX_REALTIME_JITTER_US` | `5_000` $\mu$s (5 ms) | Hard jitter SLA gate for `realtime` routes |
| `MAX_REALTIME_LOSS_PPM` | `5_000` ppm (0.5%) | Hard packet loss SLA gate for `realtime` routes |
| `PROTOCOL_VOUCHER_BURN_PPM` | `50_000` ppm (5%) | Retired in 1.1 (ADR-0053): vouchers are commercial only and carry no protocol burn |
| `MAX_AGGREGATOR_COMMISSION_PPM` | `200_000` ppm (20%) | Later phase. Maximum commission a Fleet Aggregator UID can deduct from sub-nodes |
| `SENSE_MAX_RUNS_PER_DAY` | `3` | System One runs a skill at most this often per symptom code in 24 hours, then escalates (ADR-0067) |
| `FAULT_REPORTS_PER_WINDOW` | `4` | Most fault reports a node may send per window; validators' stores drop the rest (ADR-0067) |

Peering parameters (`PEER_CHANGES_PER_WINDOW`, `PEER_EXPLORE`, `PEER_SUSPEND_WINDOWS`, the draw size and `MAX_PEERS`) are set by their ADRs and are not yet fixed. Wholesale parameters (ADR-0063, ADR-0065) are proposed; §6.2 lists the guard values, and the floor, curve and minimum purchase are calibrated before mainnet.

---

## 11. Conclusion

Transit Fabric demonstrates that an open, decentralized network does not have to choose between cryptographic verifiability and datacenter line-rate performance. By deriving every node's `/128` `fc00::/8` IPv6 identity from its Ed25519 key, flattening the control plane into end-to-end eBGP with competitive Mechanism 1 route solvers that will also plan the fabric's peering, isolating public internet egress to registered exit partners and gateways (`ADR-0004`, `ADR-0052`), enforcing conservation of flow via co-signed `HopReceipts`, paying nodes only in emissions for what validators measure, and letting buyers burn TAO for capacity with no treasury in between, Transit Fabric aims to turn idle global transit capacity into a programmable, institutional-grade backbone. Its transit and route-solving core runs today on Bittensor testnet, subnet 611.