louzt/quic-tunnel

Source: Private · **Status:** Active · **Period:** 2024-06 (active)

Multi-transport SSH fallback that survives hostile transit: layered QUIC, Hysteria2, gost, tls-direct, and direct SSH with CA pinning.

The challenge

My SSH sessions to the VPS were dropping roughly every hour on Mexican carrier paths. The carrier's traffic inspection was matching common SSH fingerprints and shaping the connection mid-session. Standard SSH tooling gave me no visibility into which transport was being shaped, and a single-transport tunnel is a single point of failure.

My role

I designed and shipped a layered transport fallback daemon that picks the best available path on connect and degrades through successive tiers when one path fails. The daemon is the transport layer my loust-vps bridge runs on.

What I did (3–5 decisions)

1. Layered transport chain across five tiers

Why: Each tier has different fingerprint properties, so when one is matched and shaped the next has a different signature. The chain runs through successively lighter transports until one survives.

Tradeoff: More code surface and more moving parts than a single-transport tunnel. The alternative was hourly session drops, so the extra surface is the price of reliability.

2. CA pinning end-to-end

Why: A downgrade attack could otherwise re-route my session to an attacker-controlled endpoint. Pinning the CA at both ends makes the session refuse any peer that isn't on the allow-list.

Tradeoff: Peer setup is a five-minute ceremony instead of one minute. The setup cost is a one-time price for a session that can't be silently redirected.

3. Connection migration for hostile transit

Why: When a middlebox disrupts a connection mid-session, the tunnel re-establishes on a fresh path without dropping the work in flight. The session stays alive through the disruption.

Tradeoff: Adds about ten milliseconds of latency on first reconnect. Saves the thirty-plus seconds of manual reconnect time I used to spend on every drop.

What changed

Session survival

Baseline: Roughly hourly drops on Mexican carrier paths

Twelve-plus hours of continuous sessions

Evidence: loust-vps bridge telemetry, 2026-Q3

Reconnect time on drop

Baseline: 30+ seconds, manual

Under 1 second, automatic

New-peer setup time

Baseline: 1 minute

5 minutes (CA pin ceremony)

Evidence: estimate, not measured

Overall tradeoffs

The daemon is private because the tier order and the fingerprinting rules are tuned to my own carrier paths. Publishing the rule order would give anyone probing my transport a roadmap; the patterns generalize, the configuration does not.

Lessons

Transport hardening is closer to plumbing than to cryptography — it is the boring operational details that determine whether a system stays up at 3am, not the cleverness of the handshake.

Stack

Go | QUIC | Hysteria2 | TLS | SSH

Evidence

- [repo] github.com/louzt/quic-tunnel