LOUST-PRO/NetBoozt_InternetUpgrade

Source: LOUST · **Status:** Active · **Period:** 2024-03 (active)

TCP and DNS tuning plus connectivity diagnostics for ISP-constrained paths: surfaces the actual transport chain, not just ping.

The challenge

Standard networking diagnostics (ping, mtr, dig) could not tell me whether a slow connection was DNS, TCP, TLS, or the application protocol. On Mexican ISP paths, deep-packet inspection was the dominant variable, and I had no way to see it.

My role

I built a desktop tool that surfaces the actual transport chain — resolver, RTT, TLS handshake — with per-ISP profile switching so the user can tune behavior for each carrier without restarting the daemon.

What I did (3–5 decisions)

1. Per-ISP profile switching

Why: Each Mexican carrier (Telmex, Totalplay, AT&T MX) has different inspection patterns, and one configuration does not fit all. Switching profiles live lets the tool follow the carrier rather than fight it.

Tradeoff: Three profiles to maintain instead of one. Each carrier gets the configuration it actually needs, not a lowest-common-denominator compromise.

2. Transport-chain diagnostics

Why: A single ping number hides the bottleneck. Per-stage numbers (DNS resolution time, TCP handshake, TLS negotiation, application roundtrip) surface exactly where the slowdown lives.

Tradeoff: More diagnostic output to read than a single latency number. Each number answers one specific question, which is the point.

3. Tauri UI over a Rust backend

Why: Native desktop feel without the Electron bloat. The diagnostic surface stays responsive even when the backend is mid-probe across multiple transports.

Tradeoff: Smaller ecosystem of Tauri UI libraries than Electron. For a single-author tool the ecosystem does not matter; for a large team it might.

What changed

Bottleneck identification

Baseline: About 10 minutes of guessing and switching tools

Under 1 minute reading the chain

Evidence: estimate, not measured

Profile switching

Baseline: Required daemon restart

Live switch in under 5 seconds

Tool startup time

Baseline: About 3 seconds (Electron baseline)

Under 1 second (Tauri)

Overall tradeoffs

I gave up the Electron ecosystem for native performance. For a tool I open daily, the speed wins; for a tool I open monthly, the ecosystem might.

Lessons

Diagnostics tools earn their keep by answering one specific question, not by being a Swiss-army knife. A focused tool that answers the bottleneck question beats a general one that does not.

Stack

Rust | Tauri | Linux | networking

Repository

github.com/LOUST-PRO/NetBoozt_InternetUpgrade

Evidence

- [repo] github.com/LOUST-PRO/NetBoozt_InternetUpgrade