LOUST-PRO/LinkMarks

Source: LOUST · **Status:** Shipped · **Period:** 2024-12 (active)

CRDT-backed bookmark manager: local-first storage, deterministic cross-device merge, no central server.

The challenge

Browser bookmark sync was cloud-locked — Pocket, Raindrop, the rest — and required a central server. I wanted bookmarks that synced across my devices without a coordinating server and with no merge conflicts when I edited on two devices between syncs.

My role

I built a CRDT-backed bookmark manager in Rust with local-first storage, opportunistic sync, and a v2.2.0 release shipped across seven LOUST-PRO crates in August 2026.

What I did (3–5 decisions)

1. CRDT reconciliation across devices

Why: Conflict-free merge means no merge-conflict recovery flow. Every device converges to the same bookmark set regardless of which edits arrived first.

Tradeoff: More storage per bookmark than a plain list, because the merge metadata is part of the record. Convergence is the prize — losing a bookmark on a sync conflict is worse than the storage cost.

2. Local-first storage

Why: Bookmarks live on disk in a portable format. Sync is opportunistic rather than authoritative, so the local copy stays usable even when the network is down.

Tradeoff: Larger local footprint than cloud-only bookmarks. The offline guarantee earns the disk space.

3. Opportunistic sync

Why: Devices on the same LAN sync fast over peer discovery; remote devices sync via an optional relay when one is available. The sync path is opportunistic, not mandatory.

Tradeoff: The relay is optional infrastructure to maintain. For a single-author project the relay is trivial; for a multi-user product it would be a real cost.

What changed

Sync conflicts

Baseline: Occasional (cloud services with last-write-wins)

None (CRDT guarantees deterministic merge)

Cross-device convergence

Baseline: Minutes (cloud roundtrip)

Under 1 second over LAN discovery

Storage portability

Baseline: Cloud-only — no local copy

On-disk portable format, sync is optional

Overall tradeoffs

I gave up the convenience of cloud-only for the offline guarantee. For research-workflow substrate, the guarantee wins — losing bookmarks on a sync conflict would be worse than the local disk cost.

Lessons

Local-first is not just an architecture choice — it is an availability guarantee. Once the substrate works without the network, the network becomes a convenience instead of a dependency.

Stack

Rust | CRDT | bookmarks | local-first

Repository

github.com/LOUST-PRO/LinkMarks

Evidence

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