NilStore Network: A Protocol for Decentralized, Verifiable, and Economically Efficient Storage

Contents

NilStore Network: A Protocol for Decentralized, Verifiable, and Economically Efficient Storage

(White Paper v2.6 - Elastic Performance)

Date: 2025-12-04 **Authors:** NilStore Core Team

Abstract

NilStore is a decentralized storage network that unifies storage and retrieval into a single **Demand-Driven Performance Market**. By treating user retrievals as valid storage proofs (**Unified Liveness**), the protocol eliminates wasted work. Placement is **System-Defined** but **Hint-Aware**. Crucially, the network supports **User-Funded Elasticity** with **8 MiB Mega-Data Units (MDUs)** and **Stripe-Aligned Scaling**, ensuring viral content remains available without punishing successful nodes.

1. Introduction

1.1 The "Double-Pay" Problem

Legacy networks treat "Storage" and "Retrieval" as separate jobs. This is inefficient. NilStore unifies them.

1.2 Key Innovations

- **Unified Liveness:** A user downloading a file *is* the storage audit.
- **Synthetic Challenges:** The network audits cold data automatically.
- **Performance Market:** Rewards are based on speed (Platinum/Gold/Silver).
- **Elasticity:** The network automatically scales replication using **Stripe-Aligned Scaling** to meet demand, funded by the user's prepaid escrow.

---

2. The Unified Liveness Protocol

2.1 Hot Data (Path A)

1. **User Request:** "I need chunk #50."
2. **Service:** SP sends data + KZG Proof.
3. **Receipt:** User signs the receipt.
4. **Consensus:** SP submits the receipt to the chain.
5. **Result:** SP earns **Storage Reward** (for proving liveness) AND **Bandwidth Fee** (for serving user).

2.2 Cold Data (Path B)

1. **System Silence:** No user asks for data.
2. **Beacon Challenge:** Chain issues "I need chunk #50" (Pseudo-random).
3. **Service:** SP computes KZG Proof.

4. **Consensus:** SP submits proof to chain.
5. **Result:** SP earns **Storage Reward** (for proving liveness).

---

3. Traffic Management (Elasticity)

3.1 The Saturation Signal

If a Platinum-tier Provider is overwhelmed by traffic, they can submit a **Saturation Signal** to the chain. * **Condition:** The SP must be in good standing (Platinum/Gold) and show high receipt volume. * **Response:** The Chain verifies the user has **Budget Available** in their escrow. * **Action:** The Chain spawns **Hot Replicas** on new Edge nodes to absorb the load. The original SP is *not* penalized.

3.2 User Controls

- **Budget Cap:** Users set a `MaxMonthlySpend`. The protocol will never spawn replicas if it would exceed this cap.
- **Result:** "Viral" content scales automatically. "Budget" content is rate-limited.

---

4. The Lifecycle of a File

Step 1: Ingestion & Placement

1. **Deal Creation:** User submits `MsgCreateDeal(Hint: "Hot", MaxSpend: 100 NIL)`.
2. **Assignment:** Chain deterministically assigns 12 SPs for **8 MiB MDUs**.
3. **Upload:** User uploads data.

Step 2: The Liveness Loop

- **Scenario 1 (Viral):** Users swarm the file. SPs signal saturation. Chain checks `MaxSpend`. Chain spawns 5 more **Stripe-Aligned** replicas.
- **Scenario 2 (Archive):** File sits idle. Chain issues Beacon challenges.

---

5. Security Analysis

Threat	Mitigation
Sybil Attack	**System-Defined Placement.**
Constraint Attack	**Rebalancing Fees.** SPs pay to rotate off shards early.
Billing Runaway	**Spend Caps.** Protocol strictly enforces user-defined limits.

---

6. Enterprise Features: Privacy & Deletion

NilStore is built for **Zero-Trust** environments.

6.1 Zero-Knowledge Cloud

- **Encryption:** Data is encrypted client-side (`AES-256-GCM`) before it ever touches the network.
- **Blind Replication:** When the network scales up "Hot Replicas," it copies **8 MiB Encrypted MDUs** as ciphertext. Providers act as blind mules; they store and serve data they cannot read.
- **Zero-Touch Scaling:** Because the encryption is deterministic, the network can replicate data autonomously. The Data Owner does **not** need to be online to re-encrypt data for new nodes.

6.2 Proof of Deletion (Crypto-Erasure)

Regulatory compliance (GDPR/CCPA) requires the ability to delete data. * **The Problem:** You cannot prove a remote server wiped a hard drive. * **The Solution:** We rely on **Crypto-Erasure**. * **Mechanism:** The User holds the **File Master Key (FMK)**. To "delete" the data globally, the User destroys the FMK. The encrypted data remaining on the network becomes mathematically irretrievable garbage.

---

7. Roadmap

1. **Phase 1:** Core Crypto & CLI (Completed).
2. **Phase 2:** Local Testnet & Specs (Completed).
3. **Phase 3:** Implementation of **Deal Object**, **Provider Capabilities**, and **System Placement**.
4. **Phase 4: Retrieval Market** & Edge SDK.