LVS — Testnet Launch Plan (EN)

Version 1.0 — Final, Developer & Investor Ready

---

1. Purpose of This Document

This document defines the **complete plan for launching the first LVS Testnet** — the point where LVS transitions from a theoretical system to a functioning distributed network.

It outlines: - technical milestones, - required components, - node implementation priorities, - network architecture, - deployment phases, - monitoring and evaluation, - success criteria.

This is the core roadmap for the first real LVS network.

---

2. Overview of LVS Testnet

The LVS Testnet is a **public experimental network** designed to: - validate Drift-Based Consensus (DBC) under real-world conditions, - measure convergence across heterogeneous nodes, - evaluate network stability under noise, loss, and partial failures, - collect performance and behavior data, - onboard early developers and researchers.

This Testnet **does not carry financial value** and is purely for engineering validation.

---

3. Testnet Architecture

The Testnet consists of three classes of nodes:

3.1 Browser Micro-Nodes (Primary)

- run in Chrome/Firefox/Edge/Safari,
- WebRTC communication,
- zero installation,
- ideal for mass participation.

3.2 Light Server Nodes (Optional)

- act as WebRTC signaling hubs,
- relay nodes for NAT-restricted peers,
- provide temporary storage for metrics.

3.3 Developer Nodes (Desktop / Server)

- Go or Rust implementation,
- used for stress tests and data logging.

The first Testnet must run **on browser micro-nodes**.

---

4. Core Components Required for Testnet

The Testnet requires minimal but complete functionality.

4.1 Peer Discovery (WebRTC Signaling)

A signaling server provides initial session setup: - Node A announces presence, - Node B receives offer, - WebRTC DataChannel establishes direct peer link.

Technology: Node.js or Go signaling server.

4.2 Message Layer

Each node must support two messages:

Entropy Packet (EP)

```
{
  type: "ep",
  entropy: [x, y],
  ts: timestamp
}
```

State Diff Message (SDM)

```
{
  type: "sdm",
  diff: [dx, dy],
  w: weight
}
```

4.3 Drift Cycle Engine

Runs every **50–150 ms**: 1. Generate entropy 2. Collect diffs 3. Compute drift 4. Update state 5. Broadcast diff 6. Render node movement on screen

4.4 Visualization Layer

A minimal UI to show: - node position (2D point), - drift direction, - peer connections, - entropy intensity.

This makes the Testnet **demonstrable**.

---

5. Testnet Phases

The Testnet rollout is divided into three phases.

Phase 1 — Private Testnet (Internal)

Goal:

Validate core drift mechanics on 2–10 nodes.

Requirements:

- Browser node MVP
- Signaling server
- Basic UI
- Logging

Tests:

- entropy generation
- drift application
- pairwise convergence
- recovery after packet loss
- behavior under latency

Expected Output:

- drift cycle validation
- basic stability metrics

Timeline: **1–2 weeks**.

Phase 2 — Closed Public Testnet (Invite-Only)

Goal:

Test drift consensus across **50–200 nodes**.

Requirements:

- optimized WebRTC mesh,
- diff throttling,
- improved UI,
- anomaly detection.

Tests:

- convergence with large peer groups
- stability under random disconnects
- recovery from node churn
- multi-region latency

Expected Output:

- convergence confirmation at medium scale
- performance baseline

Timeline: **3–6 weeks**.

Phase 3 — Open Public Testnet (Global)

Goal:

Thousands of browser nodes connecting worldwide.

Requirements:

- auto-scaling signaling layer (e.g., cloud)
- fault-tolerant message routing
- basic public dashboard
- versioned node builds

Tests:

- global convergence behavior
- resilience under load
- survival during partial region outages
- measuring emergent patterns

Expected Output:

- fully validated DBC behavior in the wild
- dataset for research paper #2

Timeline: **6–10 weeks**.

---

6. Monitoring & Data Collection

The Testnet must collect minimal, anonymous metrics: - drift vector variance - node online/offline events - packet loss ratio - convergence speed - region-to-region latency impact

No personal data. No identity.

All metrics stored as aggregated floats.

---

7. Security Considerations for Testnet

Even Testnet must enforce: - bounded drift, - clamped diffs, - anomaly rejection, - recovery mode.

This protects the network from runaway behavior.

---

8. Testnet Deployment Infrastructure

Recommended minimal infrastructure:

Signaling Servers (2–3 regions)

- EU
- US
- Asia

Static Hosting

- github.io or Cloudflare Pages for browser client

Monitoring Dashboard

- hosted on lvs.network/testnet

No blockchain nodes, no databases, no heavy infrastructure.

---

9. People & Roles

Even if one person builds the prototype, Testnet implies roles:

- **Architect / Lead Developer** — node core, drift engine
- **Frontend Developer** — browser UI, visualization
- **Backend Engineer** — signaling server
- **Research / Metrics** — analysis of drift behavior

For now, this can be **one or two people**.

---

10. Testnet Success Criteria

The Testnet is considered successful when:

1. 100+ nodes connect with no central authority

2. Drift convergence is observed across the network

3. System stabilizes under packet loss and latency

4. No single node can dominate or disrupt consensus

5. System recovers from partial network failures

6. Metrics confirm stable behavior

This proves the feasibility of LVS at scale.

---

11. After Testnet — What Comes Next

After successful Testnet completion:

1. VaultGuard Full Implementation

Advanced invariants and anomaly models.

2. Sharding & Redundancy Layer

Distributed state reliability.

3. Node Diversity

Android/iOS micro-nodes, desktop nodes, server nodes.

4. Public Developer SDK

Libraries for JavaScript, Go, Rust.

5. Research Paper #2

Based on real Testnet data.

6. Mainnet Candidate Architecture

Defines the global autonomous layer.

---

12. Conclusion

This Testnet Launch Plan defines the exact technical and operational pathway to bring LVS from prototype to a live distributed network.

With this roadmap, LVS is ready to transition from documents to reality — from theory to a functioning global autonomous value layer.