Origami Verse: Conditional Markets and Worst-World Lending on NEAR

Version 1.0

The Origami Verse team

https://multiverse-lending.vercel.app

https://github.com/skyyycodes/multiverse-near

November 2026

Abstract

Origami Verse is a protocol on NEAR that lets people hold, trade and borrow against an asset *one possible future at a time*. A market is built from one to three real-world yes/no events. Every combination of their outcomes is a *world*, so a market has two, four or eight worlds. One real token can be split into one conditional token per world; each conditional token pays out only if its world is the one that comes true. Each world has its own pool and order book, so a holder can keep NEAR in the futures they like and swap it for dollars in the futures they fear. Lending values a borrower's position in every world that is still possible and sizes the loan by the *worst* of them, so the collateral covers the debt whichever world happens. The protocol adds three lending tiers (Safe, Normal, Bold), liquidation that pays in tokens or in cash, an insurance reserve, markets that anyone can propose and fund with pledges, resolution from public sources with a dispute window and an optional run inside a trusted execution environment, and an outside NEAR/USD check taken from four public price sources. Everything runs in one Rust contract on NEAR testnet. This paper explains the model, the maths and the implementation in plain words, with the real parameters read from the contract and real transactions from the testnet deployment.

Contents

1 Introduction 2

2 The model: events, worlds and verse tokens 3

3 Trading 4

4 Prices for lending 6

5 Conditional lending 7

6 Market creation by proposals and pledges 10

7 Resolution 12

8 The outside NEAR/USD reference 14

9 Implementation on NEAR 15

10 Security and invariants 16

11 Deployment and evaluation on testnet **17**

12 Limitations and future work **19**

13 Conclusion **19**

1 Introduction

1.1 The problem

Suppose you hold NEAR and you have a specific worry: "If the next big upgrade slips *and* the US central bank does not cut rates, NEAR will fall." Today you have two blunt tools. You can sell NEAR, or you can buy a put option. Both protect you in *every* future, including the futures you expect to go well, and you pay for that protection everywhere. A prediction market lets you bet on one event, but it does not let you say "keep my NEAR if the upgrade ships, turn it into dollars if it does not". A lending market does not accept such a position as collateral at all.

Origami Verse fixes this by cutting an asset along the lines of possible futures. You protect yourself only in the futures you fear, you keep your upside in the others, and you can still borrow real dollars against the result.

1.2 The idea in one picture

Take two real events:

- Event 1: "Will SPICE ship on NEAR mainnet by Mar 31, 2027?" Outcomes: SHIPS or DELAYED.
- Event 2: "Will the Fed cut at its December 8–9, 2026 meeting?" Outcomes: CUTS or HOLDS.

Together they describe four possible futures, which we call *worlds*: SHIPS·CUTS, DELAYED·CUTS, SHIPS·HOLDS and DELAYED·HOLDS. Exactly one of them will come true. One wNEAR (wrapped NEAR, a normal token) can be split into four pieces: "wNEAR if SHIPS·CUTS", "wNEAR if DELAYED·CUTS", and so on. Each piece pays one real wNEAR if its world comes true and nothing otherwise. Holding all four is the same as holding one wNEAR, because exactly one of them will pay.

The idea of splitting an asset into one version per possible outcome and moving value between those versions comes from Paradigm's essay "Multiverse Finance" [1]. Origami Verse is our own design and implementation of that idea on NEAR, with trading, lending, market creation and resolution built around it.

1.3 What this paper covers

Section 2 defines events, worlds, masks and conditional tokens, and the rules that keep every token fully backed. Section 3 explains the pools, the odds and the order book. Section 4 shows how lending reads prices safely. Section 5 covers worst-world lending, tiers, interest, liquidation and bad debt. Section 6 explains how anyone can propose and fund a new market. Section 7 covers resolution, and Section 8 the outside NEAR/USD check. Section 9 describes the NEAR implementation, Section 10 security and invariants, Section 11 the live testnet deployment, and Section 12 what is still missing.

Where the numbers come from. Every parameter in this paper was read from the contract source (`contracts/multiverse/src` and `crates/multiverse-math/src`) or from the live contract `app.ifverse-staging.testnet`[1] on NEAR testnet on 14 November 2026. Prices and balances from the live markets are snapshots: they move as people trade.

2 The model: events, worlds and verse tokens

2.1 Events and outcomes

An **event** is a yes/no question about the real world with two named outcomes, a written **rule** that says exactly how it resolves, and a public **source**. For example, event 2 on the live contract has the rule: "CUTS if both limits of the federal funds target range (FRED DFEDTARU and DFEDTARL) on 2026-12-10 are lower than on 2026-12-08; HOLDS if neither is lower (a hike resolves HOLDS)." Each rule ends with a short machine tag such as `[ifverse:fomc:2026-12-08:2026-12-10]`, which tells the off-chain tools what data to read. Rules are stored on chain (up to 1,024 bytes).

2.2 Worlds and masks

A **market** (called a *scenario* in the code) combines n events, with $1 \le n \le 3$. A **world** is one combination of outcomes, so a market has $W = 2^n$ worlds: 2, 4 or 8. Worlds are numbered $0, \ldots, W-1$. In world w, event k has outcome

$$o_k(w) = \left\lfloor w/2^k \right\rfloor \bmod 2, \tag{1}$$

where outcome 0 is the first named outcome (SHIPS, CUTS, YES) and outcome 1 the second.

In plain words. Write the world number in binary. Bit 0 is the answer to the first event, bit 1 the answer to the second event, bit 2 the answer to the third. World 2 is binary `10`: the first event took outcome 0 (SHIPS), the second took outcome 1 (HOLDS), so world 2 is SHIPS·HOLDS.

A **mask** is a set of worlds, written as a number whose bit w is set when world w is in the set. In the four-world market, mask 1 (`0001`) is SHIPS·CUTS alone, mask 3 (`0011`) is "the Fed cuts" (worlds 0 and 1), mask 5 (`0101`) is "SPICE ships" (worlds 0 and 2), and mask 14 (`1110`) is every world except SHIPS·CUTS. The mask with every bit set is the **root**: it means "in every world", which is just the real asset. A four-world market has 15 non-empty masks and an eight-world market 255.

Each market also keeps a mask called `possible`: the worlds that can still happen. It starts as the root and loses worlds as events resolve.

2.3 Verse tokens

A **verse token** is one asset on one mask of one market. "wNEAR-if-mask-14 in market 1" pays one real wNEAR if the world that comes true is in mask 14. Origami Verse supports two assets, wNEAR and USDC, so in every market each non-root mask has a wNEAR verse and a USDC verse. Each verse is a NEP-245 multi-token [6] with the id `{market}:{asset}:{mask}`, for example `1:wnear:14`.

A verse is *live* while its mask still contains a possible world, *dead* once it does not, and *collapsed* once its mask contains every possible world. A collapsed verse is as good as the real asset: it redeems one for one.

[1] The testnet accounts (`*.ifverse-staging.testnet`), the rule tags and the web address keep earlier working names; the project was formerly IfVerse / Multiverse Lending.

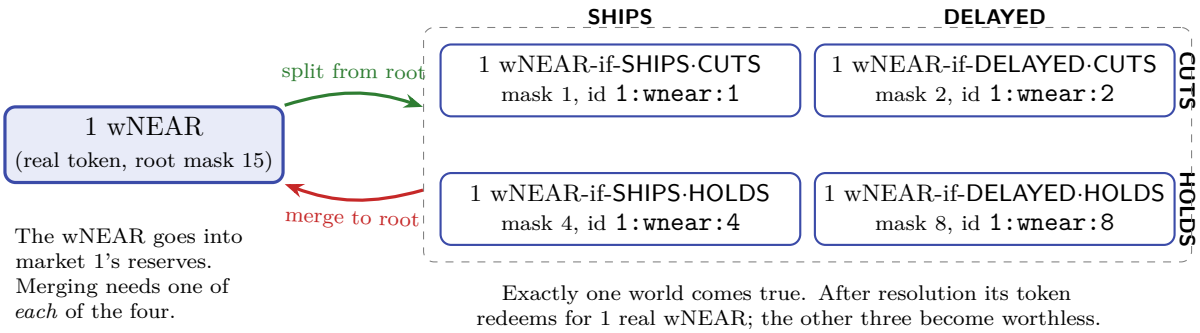


Figure 1: Splitting one wNEAR in a market of two events (four worlds). Columns are the outcomes of event 1 (SPICE), rows the outcomes of event 2 (the December Fed meeting). Splitting and merging never change the total backing.

2.4 Split, merge and redeem

Three operations move value between masks. None of them uses a price.

- **Split** turns an amount on one mask into the same amount on each part of a partition of that mask. The parts must not overlap and together must cover every possible world of the parent. Splitting from the root takes real tokens in and puts them in the market's **reserves**.
- **Merge** is the reverse: equal amounts of all the parts become the parent. Merging into the root pays real tokens out of the reserves.
- **Redeem** pays one real token for each collapsed verse token after resolution.

Split and merge only burn and mint, one for one, with no rounding. Reserves move only on a split from the root, a merge into the root and a redeem. This gives the central rule of the system, the **backing invariant**: for every market, every asset and every possible world w,

$$\sum_{\text{masks } M \ni w} \text{supply}(M) \;=\; \text{reserves}\,. \tag{2}$$

In plain words. Pick any world that could still happen. Add up every token that would pay out if that world came true. The total always equals the real tokens locked in the market. So whichever world wins, there is exactly enough real money to pay everyone who holds a winning token. No price, pool or loan can break this, because they only move existing tokens between people.

3 Trading

Splitting by itself only changes the shape of what you hold. To *hedge*, you need to swap: give up NEAR in the worlds you fear and get dollars there instead. Origami Verse offers three kinds of pool and an order book. Internally every amount is an 18-decimal integer; prices are USDC per wNEAR; fees are counted in *pips* (1,000,000 pips = 100%).

3.1 World pools: the price of NEAR in one world

A **world pool** trades wNEAR-if-w against USDC-if-w for a single world w. Both sides pay only in world w, so the pool's price is "what NEAR is worth if world w happens". Pools are constant-product pools with the behaviour of a full-range concentrated-liquidity position [10]:

$$x \cdot y = L^2, \qquad p_w = \frac{y}{x}, \tag{3}$$

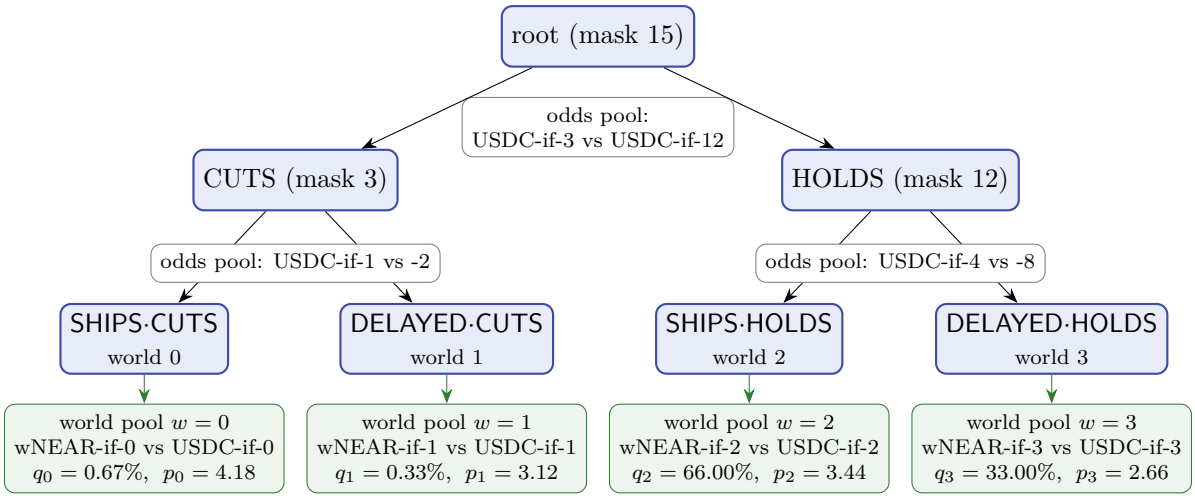


Figure 2: Market 1's pools: three odds pools (grey) arranged as a tree give each world its probability q_w, and four world pools (green) give each world its NEAR price p_w. Seven pools in total, which is the `pool_count` the contract reports for this market. Values as read on 14 Nov 2026.

where x is the wNEAR-if-w in the pool, y the USDC-if-w, L the pool's liquidity and p_w its price.

In plain words. The pool keeps the product of its two balances fixed. When you sell NEAR into it, x goes up, so y must go down: you get USDC out, and the price of NEAR in that world falls a little. A bigger pool (larger L) moves less for the same trade.

On 14 November 2026 the four world pools of market 1 priced NEAR at 4.18 (SHIPS·CUTS), 3.12 (DELAYED·CUTS), 3.44 (SHIPS·HOLDS) and 2.66 (DELAYED·HOLDS) USDC. The market believes NEAR is worth most if SPICE ships and the Fed cuts, and least if SPICE slips and the Fed holds.

3.2 Odds pools: how likely each world is

An **odds pool** trades USDC-if-X against USDC-if-Y for two disjoint masks X and Y. Because one dollar in X and one dollar in Y only differ in *when* they pay, their exchange rate is the market's view of how likely X is compared with Y. A pool with reserves r_X and r_Y gives X the share

$$s_X = \frac{r_Y}{r_X + r_Y} \tag{4}$$

of their common parent. Odds pools form a tree from the root down to single worlds (Figure 2). A world's probability q_w is the product of the shares along its branch. The second child of each split takes "parent minus first child", so the probabilities always add up to exactly 1. If a split between two live masks has no pool, the odds are reported as unknown rather than guessed.

In plain words. If USDC-if-CUTS is plentiful in its pool and USDC-if-HOLDS is scarce, "a dollar if the Fed cuts" is cheap and "a dollar if the Fed holds" is dear: traders think a cut is unlikely. The pool's balances turn that view into a percentage.

3.3 The implied spot price

If we know each world's probability q_w and each world's NEAR price p_w, the price of plain NEAR today should be their weighted average:

$$S_{\text{implied}} = \sum_{w \in \text{possible}} q_w\, p_w. \tag{5}$$

In plain words. Today's NEAR is "NEAR in whichever world happens". Its fair price is each world's price, weighted by how likely that world is.

For market 1 on 14 November 2026 this gives $0.0067 \cdot 4.18 + 0.0033 \cdot 3.12 + 0.6600 \cdot 3.44 + 0.3300 \cdot 2.66 \approx 3.19$ USDC per NEAR. (The SPICE odds in this market are an assumption of 66.67% because no public market exists for that event; the Fed odds were seeded from a dated reading and are labelled as such on chain.) Section 8 compares this number with the real NEAR price from exchanges.

3.4 Fees

World and odds pools charge a **dynamic fee**. Each pool remembers the signed direction of its last 8 swaps. If the recent flow has net size $|net|$ out of a total $total$, the next swap pays

$$\text{fee} = \text{base} + \left\lceil (\text{cap} - \text{base}) \cdot \frac{|net|}{total} \right\rceil, \qquad \text{base} = 0.30\%,\ \text{cap} = 1.50\%. \tag{6}$$

In plain words. When trades go both ways, everyone pays the normal 0.30%. When everyone pushes in the same direction (for example, many people dumping NEAR in a feared world), the fee climbs towards 1.50%. One-sided rushes pay more, which protects the liquidity providers.

The fee is taken from the input and kept outside the curve for liquidity providers. There is also an optional **spot pool** for real wNEAR against real USDC with a fixed 0.30% fee.

3.5 The order book

Next to the pools, every live non-root mask has a limit-order book. A *bid* offers to buy wNEAR-if-mask at a price and locks USDC-if-mask as escrow; an *ask* offers to sell wNEAR-if-mask and locks it. Rounding always favours the side that is owed: whoever pays USDC pays the rounded-up amount, whoever is paid from a bid's escrow gets the rounded-down amount, so an open bid always covers every fill that is still possible.

Orders fill only in the *epoch* they were posted in (the market's resolution count), before their expiry, while their mask is live, and while no outcome proposal is open. Limits from the code: at least 0.001 wNEAR and 0.0001 USDC of value per order, and at most 64 open orders per account.

3.6 The router: one signature for a whole hedge

The **router** turns a hedge into one call. `conditional_sell` takes wNEAR, splits it from the root into every possible world, and in each world you chose fills planned orders first and sends the rest into the world pool. Each world must return at least a minimum output, or the whole call fails. The result stays in your account or, if you ask, becomes loan collateral. `conditional_buy` is the mirror image with USDC, and `sell_and_borrow` sells and then borrows USDC in the same call. A trade may use at most 20 order fills in markets of up to four worlds and 16 in eight-world markets.

4 Prices for lending

A lending protocol is only as safe as the prices it reads. A pool's current price can be pushed by one large trade, so lending never reads it alone. Origami Verse uses three defences: a time-weighted average, a set of health checks per pool, and a comparison with the outside NEAR/USD price (Section 8).

4.1 Time-weighted average price

Every pool keeps a ring of up to 128 observations of its cumulative tick (a tick is a logarithmic price step). The **TWAP** (time-weighted average price) over a window T is

$$\bar{p}_w = 1.0001^{\,\bar{t}}, \qquad \bar{t} = \frac{C(t_{\text{now}}) - C(t_{\text{now}} - T)}{T}, \tag{7}$$

where C is the cumulative tick-seconds. At most one observation counts per second, so a trade and its reversal in the same block do not move the average. On the live contract the default window for lending is $T = 1{,}800$ seconds (30 minutes).

In plain words. Instead of asking "what is the price this second?", lending asks "what was the average price over the last half hour?". To fool it, an attacker would have to hold a false price for a long time and pay fees the whole while.

4.2 Is a world's price trustworthy?

Before lending uses a world pool, the contract sorts it into one of five states, checked in this order:

1. **no pool**: the world has no world pool;
2. **young**: the pool is younger than the minimum age (1,800 s on the live markets);
3. **shallow**: the pool holds less than the minimum depth of USDC at its TWAP price (50,000 USDC on the live markets);
4. **out of band**: the current tick is more than 953 ticks (about 10%) away from the TWAP;
5. **trusted**: none of the above.

For new borrowing, only *trusted* worlds get a price; any other world counts as zero. For liquidation, trusted worlds and out-of-band worlds that have a full TWAP window are priced at the TWAP; a young, shallow or missing world pauses liquidation in that market (it would be unfair to liquidate someone using no price at all).

4.3 Borrow side and liquidation side

The two sides read prices differently on purpose:

$$p_w^{\text{borrow}} = \min\!\left(p_w^{\text{spot}},\ \bar{p}_w\right) \cdot f_b, \tag{8}$$

$$p_w^{\text{liq}} = \bar{p}_w \cdot f_l. \tag{9}$$

Here $f_b \le f_l \le 1$ are factors from the outside reference check (Section 8); when the reference agrees with the pools they are both 1.

In plain words. When you borrow, the protocol takes the *lower* of the current price and the half-hour average, so a quick pump cannot unlock a bigger loan. When it decides whether to liquidate you, it uses only the slow average, so a quick dump cannot liquidate you. Because the borrow price is never above the liquidation price, a loan that was just allowed can never be liquidatable in the same block.

5 Conditional lending

5.1 Collateral is valued in every world

Each market has lending pools of real USDC. A borrower posts **collateral pieces**: live verse tokens of that market, or real wNEAR or USDC (up to 20 pieces per account and market). Each piece is valued in every possible world:

	Safe	Normal	Bold
Borrow LTV β (most you can borrow)	50%	60%	70%
Liquidation LTV λ	60%	70%	80%
Liquidation bonus b	5%	6%	8%
Base rate (APR)	1%	2%	3%
Slope up to the kink (APR)	4%	6%	9%
Slope above the kink (APR)	45%	52%	63%
Kink (utilization)	80%	80%	80%
Utilization cap	90%	90%	90%
Reserve factor (to insurance)	10%	10%	10%
Shared by all tiers			
Minimum loan	100 USDC		
Close factor (most of the debt one liquidation repays)	50%		
Full liquidation allowed when	HF $<$ 0.90 or debt $<$ 200 USDC		
Collateral pieces per account and market	20		

Table 1: Lending tiers as stored on the live contract (`get_app_config` and `tier_config` for market 1). Rates are stored per second in 18-decimal units; for example the Safe base rate is 317,097,919 per second, which is 1% a year.

- wNEAR-if-M of amount a is worth $a \cdot p_w$ in each world w of M, and nothing elsewhere;
- USDC-if-M of amount a is worth a in each world of M, and nothing elsewhere;
- real wNEAR is worth $a \cdot p_w$ in every world, and real USDC is worth a in every world.

The account's value in world w, written V_w, is the sum over its pieces. Its **worst-world value** is

$$C_{\min} = \min_{w \in \text{possible}} V_w. \tag{10}$$

In plain words. We ask "if this world comes true, what is your collateral worth?" for every world that can still happen, and keep the smallest answer. That is the amount the loan can rely on, whatever happens.

5.2 Health factor and borrow limit

Each lending tier has a **liquidation LTV** (λ, loan-to-value at which liquidation starts) and a lower **borrow LTV** (β, the most you can borrow). The **health factor** is

$$\text{HF} = \frac{C_{\min}^{\text{liq}} \cdot \lambda}{\text{debt}}, \tag{11}$$

where $C_{\min}^{\text{liq}}$ uses the liquidation-side prices. A new borrow, or a withdrawal of collateral, must leave

$$\text{debt} \;\le\; C_{\min}^{\text{borrow}} \cdot \beta, \tag{12}$$

using the borrow-side prices. A loan with HF < 1 can be liquidated.

In plain words. Health factor above 1 means safe; below 1 means anyone may step in and repay part of your loan in exchange for part of your collateral. Because β is 10 points below λ, a fresh loan starts with a cushion.

Resolving an event only *removes* worlds from the possible set. The minimum over fewer worlds can only be equal or higher, so **resolution never lowers a health factor**. This is checked as a property test (Section 10).

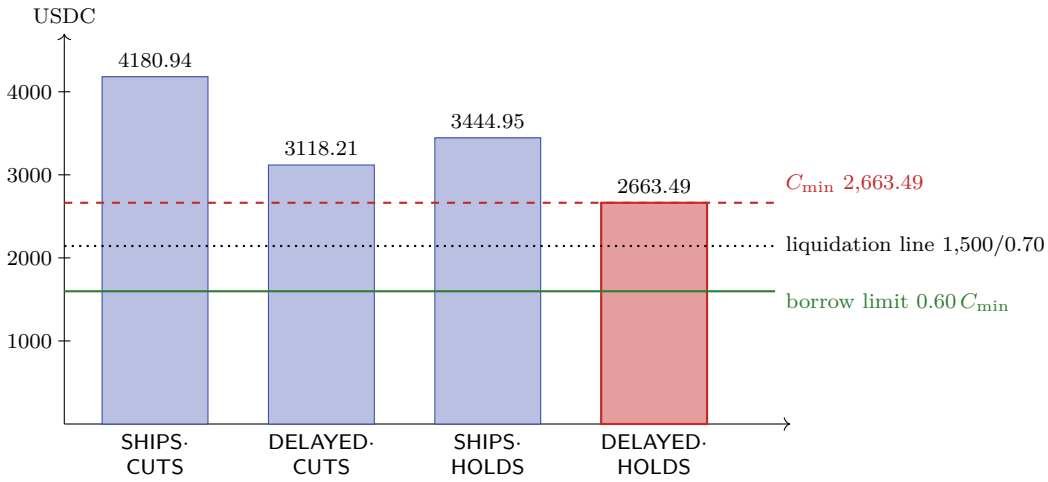


Figure 3: Worst-world valuation of 1,000 wNEAR posted as collateral in market 1 (Normal tier), at the world prices of 14 Nov 2026. The loan is sized by the lowest bar (DELAYED·HOLDS). With 1,500 USDC borrowed, liquidation starts only if that bar falls below the dotted line.

5.3 A worked example

Use market 1's world prices from Figure 2 and, to keep it simple, assume the borrow and liquidation prices are both equal to those prices. A borrower posts 1,000 real wNEAR in the Normal tier. Its value per world is

$$V = (4{,}180.94,\ 3{,}118.21,\ 3{,}444.95,\ 2{,}663.49)\ \text{USDC},$$

so $C_{\min} = 2{,}663.49$ USDC, in world DELAYED·HOLDS. The borrow limit is $0.60 \times 2{,}663.49 = 1{,}598.09$ USDC. If the borrower takes 1,500 USDC, the health factor is $2{,}663.49 \times 0.70/1{,}500 \approx 1.24$. The loan becomes liquidatable when $C_{\min}$ falls below $1{,}500/0.70 \approx 2{,}142.86$, that is, when NEAR's price in DELAYED·HOLDS falls below about 2.14 USDC, roughly 19.5% below its level in the example.

Now suppose the borrower had *hedged*: sold some wNEAR-if-DELAYED·HOLDS for USDC-if-DELAYED·HOLDS before posting. In that world part of the collateral is now dollars, so a fall in NEAR's price there hurts less. This is the point of Origami Verse: the hedge you buy for the future you fear directly raises how much you can borrow and how safe the loan is.

5.4 Interest

Each tier is its own pool of lenders' USDC with its own interest curve. The **utilization** is $u =$ borrowed/supplied (capped at 1). The borrow rate per second follows a "kinked" line:

$$r(u) = \begin{cases} r_0 + s_1 \dfrac{u}{k}, & u \le k, \\ r_0 + s_1 + s_2 \dfrac{u-k}{1-k}, & u > k, \end{cases} \tag{13}$$

with base r_0, slopes s_1, s_2 and kink k from Table 1. Interest over an interval of Δt seconds is $B \cdot r \cdot \Delta t$ on the borrowed amount B, and it compounds every time the market is touched. A share $\rho_f = 10\%$ of all interest (the **reserve factor**) goes to the tier's insurance reserve; lenders receive the rest, so the supply rate is

$$r_{\text{supply}} = r(u)\,(1 - \rho_f)\,u. \tag{14}$$

In plain words. When few people borrow, money is cheap. As the pool fills up, the rate rises slowly; past 80% it rises steeply, which pulls in new lenders and pushes borrowers to repay. Borrowing stops at 90% utilization so lenders can always withdraw some cash.

Live example: at 16:39 UTC on 14 November 2026 the Safe tier of market 1 had 105,050 USDC supplied and 1,330.56 USDC borrowed, a utilization of 1.27%. Equation (13) gives $1\% + 4\% \times 0.0127/0.80 \approx 1.063\%$ a year, which is exactly the 337,179,725 per second the contract reported.

Lender shares use virtual offsets (10^6 shares and 1 unit of assets) so the first depositor cannot inflate the share price and steal from later depositors.

5.5 Liquidation

When HF < 1 anyone can liquidate, unless the market is halted by an open outcome proposal (resolution can only raise health factors, so liquidating during the dispute window would front-run it) or a world has no price. The liquidator repays r USDC, at most

$$r_{\max} = \begin{cases} \text{debt}, & \text{HF} < 0.90 \text{ or debt} < 200 \text{ USDC}, \\ 0.5 \cdot \text{debt}, & \text{otherwise}, \end{cases} \tag{15}$$

and receives the *same fraction* f of every collateral piece:

$$\text{target} = r\,(1+b), \qquad f = \min\left(1,\ \frac{\text{target}}{C_{\min}}\right). \tag{16}$$

In plain words. The liquidator pays back part of the loan and gets that value plus a bonus (5%, 6% or 8%) in collateral. Taking the same slice of every piece keeps the borrower's worst world the worst one, so the slice is worth at least the target in *every* world. The liquidator is paid whatever happens.

A repayment may not leave a remainder between zero and the 100 USDC minimum loan, so no dust loans are left behind.

Tokens or cash. A liquidator can take the slice as tokens, or ask for **cash**: in the same transaction the contract turns the pieces into single-world tokens, sells each world's wNEAR in its world pool, merges the USDC-if-w back into real USDC as far as every world allows, and pays it out. The liquidator sets a minimum amount of cash; the sale also may not leave any world without a liquidation price. If either check fails, the whole transaction is reverted.

5.6 Insurance and bad debt

If a liquidation takes every piece and some debt is still left, that remainder is **bad debt**. It is written off in this order:

$$\text{cover} = \min(\text{bad},\ \text{insurance}), \qquad \text{loss to lenders} = \text{bad} - \text{cover}. \tag{17}$$

In plain words. The tier's insurance reserve pays first. Only what it cannot cover is shared by the lenders of that tier, all at once, through a lower share price.

The insurance reserve grows from the reserve factor and from anyone who adds USDC to it with `fund_insurance`. It is never lent out. On the live contract the Bold tier of market 1 holds about 1,500 USDC of insurance. If a market's lenders are wiped out, the market freezes.

6 Market creation by proposals and pledges

Markets do not have to be created by the team. Anyone can propose one, and the community funds its starting liquidity with **pledges**. A pledge is wNEAR or USDC held in escrow for a

proposal: it becomes the new market's pool liquidity when the market opens, and it is returned if the proposal ends without opening.

6.1 Two kinds of proposal

- **Kind A** (`propose_event`): a brand-new yes/no event with its question, rule, source, deadline, starting odds and a price multiplier per outcome (each multiplier between 0.5 and 2). The owner must approve the event; a market on it then opens when pledges reach the goal.
- **Kind B** (`propose_market`): a market on one to three events that already exist and are still open. No approval is needed: the market *opens by itself* the moment pledges reach the goal.

6.2 Pricing a new market

A new market needs starting prices. Origami Verse never takes them from a proposer's guess alone:

1. **Spot reference** S_{ref}. Read from the TWAP of one or more anchor markets (market 1 on the live contract). Anchors must agree within 3%. Because `require_external` is on, S_{ref} must also be within 10% of an outside NEAR/USD reference at most an hour old.
2. **Shape.** For Kind A, the proposer's odds q_0, q_1 and multipliers m_0, m_1, normalised so that $q_0 m_0 + q_1 m_1 = 1$. For Kind B, each event's odds and price ratios are read from its *reference market* (which must have at least 10,000 USDC of depth and agree with S_{ref} within 10%); the odds are clamped to $[1\%, 99\%]$ and combined:
$$q_w = \prod_k c_k(o_k(w)), \qquad m_w = \prod_k R_k(o_k(w)), \tag{18}$$
where $c_k(o)$ is the clamped chance of outcome o of event k and $R_k(o)$ the ratio of that outcome's price to its reference level.
3. **World prices.**
$$p_w = S_{\text{ref}} \cdot \frac{m_w}{\sum_v q_v m_v}, \tag{19}$$
so that the new market's implied spot (5) equals S_{ref}. For Kind B every world price must lie between 0.2 and 5 times S_{ref}.

In plain words. A new market starts at today's real NEAR price, spread across its worlds in the proportions that existing markets (or, for a brand-new event, the proposer's stated view within limits) already show. A proposer cannot open a market at silly prices.

6.3 The goal

The **goal** is the value of pledges needed to give every world pool a minimum depth. The planner computes the wNEAR A^* and USDC U^* needed for every world pool to hold 15,000 USDC of depth (odds pools get 25% of that depth), then

$$\text{goal} = \max\Big(10{,}000,\ \big\lceil A^* S_{\text{ref}} + U^* \big\rceil_{1,000}\Big) \text{ USDC}, \tag{20}$$

where $\lceil \cdot \rceil_{1,000}$ rounds up to a multiple of 1,000. Live goals on testnet range from 39,000 USDC (a two-world market) to 66,000 USDC (the eight-world market on three events). Pledges count at their value; a single pledge must be at least 10 USDC or 3 wNEAR.

Parameter (live value, `market_params`)	Value
Bond, Kind A / Kind B	0.2 / 0.05 NEAR
Opening reserve for 2 / 4 / 8 worlds	0.10 / 0.15 / 0.25 NEAR
Proposal lifetime while pledging	60 days
Owner review window (Kind A)	14 days
Minimum pledge window before a new event's deadline	1 day
Minimum market life after opening	3 days
Notice before opening after the owner changes terms	1 hour
Seed vault lock after opening	7 days
Depth target per world pool / minimum opening depth	15,000 / 10,000 USDC
Minimum goal, goal rounding	10,000 / 1,000 USDC
Odds pool depth relative to world pools	25%
Odds clamp	1%
Anchor agreement / reference agreement / outside agreement	3% / 10% / 10%
Minimum pledge	10 USDC or 3 wNEAR
Live proposals per account / pledges per account / live proposals per event	4 / 32 / 16
Spam floor (evictable after 3 days if pledges are below this share of the goal)	10%

Table 2: Market-creation parameters on the live contract.

6.4 Opening, refunds and the seed vault

When the pledged value reaches the goal, the market opens in one transaction: it creates the event (Kind A), the market, every world pool and odds pool at the planned prices, and a **seed vault** that holds the pools on behalf of the pledgers. Each pledger receives vault shares in proportion to what the opening used of their pledge, and gets back what it did not use. Vault shares unlock 7 days after opening. If the pledge that reaches the goal carries enough gas, it schedules the opening itself (`auto_open`); anyone can also call `open_market`. Before a market opens, a pledger can withdraw at any time.

6.5 Bonds, reserves and spam limits

A proposer attaches NEAR: a **bond** (0.2 NEAR for Kind A, 0.05 NEAR for Kind B) and an **opening reserve** (0.05 NEAR plus 0.025 NEAR per world: 0.10, 0.15 or 0.25 NEAR) that pays for the new market's storage. The Kind A bond comes back when the owner approves the event; if the owner rejects it, the owner may keep the bond for the protocol's storage fund. A combination's bond goes to the storage fund if its proposer withdraws it. The reserve comes back to the proposer, minus whatever an opening spends.

To stop spam, each account may have at most 4 live proposals and each event at most 16. A proposal that is at least 3 days old and has pledges below 10% of its goal (the **spam floor**) can be evicted to make room for a new one.

7 Resolution

7.1 From a public source to a final outcome

An event's outcome becomes final in one of three ways:

1. **Reporter.** A dedicated reporter account proposes the outcome with `report_outcome`, together with up to 512 bytes of evidence, which is logged on chain. The reporter's tools sign with a NEAR *function-call key*: it can call only `report_outcome` and `finalize` on the contract and cannot move funds.

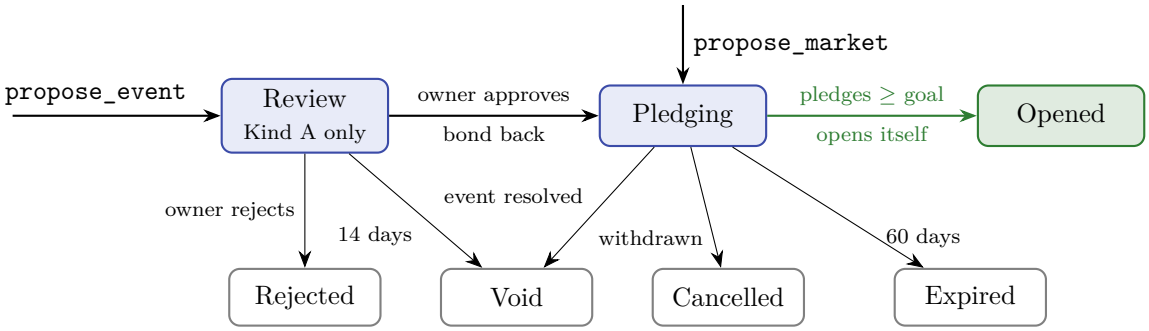


Figure 4: Proposal lifecycle. Kind A starts in Review; Kind B starts in Pledging. Pledges are accepted in Review and Pledging. The owner can also reject a new event while it is pledging, and a proposer can withdraw a proposal nobody has pledged to in either state (simplified here). Every grey end state refunds all pledges. Status changes are applied lazily by the first call that needs them, so no timer is required on chain.

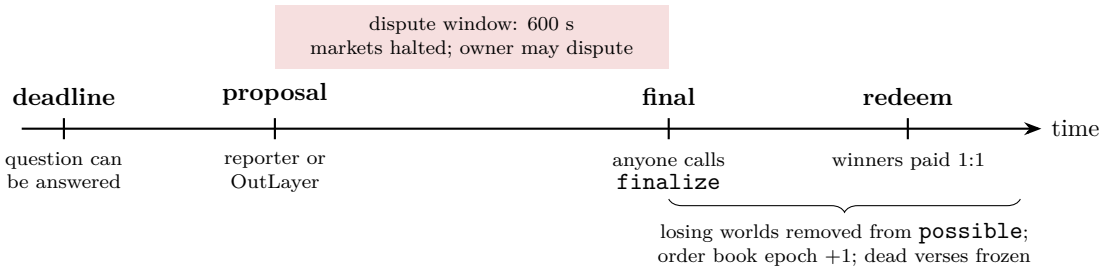


Figure 5: Resolution timeline of one event.

2. **OutLayer.** Anyone can call `request_resolution` and pay OutLayer's fee. OutLayer [9] runs a small pinned program inside a trusted execution environment (an Intel TDX worker) and hands the answer back to the contract through NEP-519 yield/resume [8]. The program's download URL and its SHA-256 hash are pinned on chain, so a different program cannot answer. A valid answer becomes an ordinary proposal; a failed run or malformed answer is logged and dropped.
3. **Owner.** `admin_resolve` finalizes at once. This is an emergency path and is trusted.

7.2 The dispute window

A proposed outcome is not final straight away. It stays open for the **dispute window**: 600 seconds (10 minutes) on the live contract. While it is open, every market that uses the event is *halted*: swaps in its pools, order fills and liquidations stop, so nobody can trade on the answer before it is final. During the window the owner may `dispute`, which cancels the proposal (a new one may follow). After the window, *anyone* can call `finalize`.

7.3 Finalize and redeem

Finalizing an event resolves every market that uses it. Worlds whose outcome for that event differs from the final one are removed from `possible`. The market's resolution count goes up, which closes every resting order of the old epoch. Verses whose masks lost all their worlds are dead and can no longer move. Once every event of a market is final, exactly one world is left; the verse tokens covering it are collapsed and redeem one for one into real wNEAR or USDC. Borrowers can close their loan and redeem in a single call (`close_after_resolution`).

7.4 Examples on testnet

- Event 3, "Did the Fed cut at its October 27–28, 2026 meeting?", was resolved through OutLayer. The pinned program reads the FRED series DFEDTARU and DFEDTARL [12]: the upper and lower limits were 4.00% and 3.75% both before and after the meeting, so the answer is HOLDS. The chain shows the event final with outcome HOLDS and `proposed_by: "outlayer"`. On the first deployment the same program ran in 1,655 ms and OutLayer charged 0.0011655 NEAR of the 0.005 NEAR deposit, refunding the rest.
- Event 4, "Did NEAR close above $4.00 on Nov 1, 2026?", was reported by the reporter from the Coinbase daily close (4.284 USDC, with Kraken's 4.2827 as supporting evidence) and finalized after the 10-minute window. Outcome: YES.

8 The outside NEAR/USD reference

Lending prices come from Origami Verse's own pools. If those pools drift away from the real world (for example because nobody trades them on testnet), loans could be sized on stale prices. Origami Verse therefore keeps an outside **reference price** R for NEAR/USD.

8.1 How the reference is formed

The reference is the *median* of NEAR/USD from four public sources: Coinbase, Kraken, Gemini and CoinGecko. It can be written by a pinned OutLayer program (`oracle/nearusd-guest`) or by the reporter key. Every new value, from either path, must have been observed at most 300 seconds before it arrives and no more than 60 seconds in the future, must be newer than the stored one, and must lie between 0.1 and 1,000 USDC. For a value from the OutLayer program the contract also checks that at least 3 sources answered and that they agree within 2% (200 bps). The reporter path sends only the price and its time, so the keeper applies the same 3-source, 200 bps rule off chain before it posts.

Example: the seed price of the live markets was the median of Coinbase 3.3204, Kraken 3.3230, CoinGecko 3.33 and Gemini 3.3477 at 04:32 UTC on 14 November 2026. With four sources the median is the mean of the middle two: $(3.3230 + 3.33)/2 = 3.3265$, with a spread of 83 bps.

8.2 How lending uses it

For each market with the check switched on, the contract compares the reference with the implied spot of (5) computed from TWAPs:

$$\rho = \frac{R}{S_{\text{implied}}}, \qquad f_b = \min(\rho, 1), \qquad f_l = \begin{cases} 1, & \rho \geq 1, \\ \max(\rho,\ 1-h), & \rho < 1, \end{cases} \tag{21}$$

with maximum haircut $h = 20\%$ on the live markets. Then:

- **New borrowing is blocked** if the reference is older than 3,600 seconds, or if $|\rho - 1| > 10\%$.
- If the pools sit *above* the real price ($\rho < 1$), both sides are scaled down by ρ (the liquidation side by at most 20%), using a reference at most 300 seconds old for liquidation.
- If the pools sit more than 10% *below* the real price ($\rho > 1.1$), liquidation waits (the *guard*): it would be unfair to liquidate someone on prices the outside world does not confirm.
- Repaying, adding collateral, withdrawing supply and exiting never wait on the reference.

In plain words. The protocol checks its own prices against real exchanges. If its pools say NEAR is worth much more than the exchanges do, it lends less. If they disagree too much, it stops new loans until they agree again. It never traps anyone: you can always repay and leave.

8.3 Limits of the check

The reference covers *today's* NEAR price only. It cannot say what NEAR *would* be worth in a particular world, so it checks the weighted average (5), not each world price. It depends on the odds being right; on the live markets one set of odds is an assumption (Section 3). On the testnet deployment the reference is written by the reporter path; the OutLayer program for NEAR/USD is built and its hash is fixed, but it is not yet configured on the live contract.

9 Implementation on NEAR

9.1 One contract

On NEAR, a call to another contract is asynchronous: it runs in a later receipt, and if it fails, the caller is not rolled back [2]. A hedge that splits, fills orders, swaps in several pools, posts collateral and borrows would need a callback and a compensation path at every step if each part were a separate contract. Origami Verse puts pools, order book, lending, proposals and the liquidity vault in *one* contract, so such a hedge is one synchronous call that either fully succeeds or fails and changes nothing. The only calls that leave the contract are payouts of real tokens (after every state change), the NEP-245 transfer-call receiver, NEAR refunds and OutLayer requests.

The contract is written in Rust with `near-sdk` 5.14 [3]. The maths lives in a separate crate with no NEAR dependency: fixed-point arithmetic with 256- and 512-bit intermediates, tick maths ported from Uniswap v4's TickMath [11], constant-product pools, the dynamic fee, TWAP, odds, shares, order maths, the opening planner and worst-world risk. Rounding always favours the protocol: what a user receives rounds down, what a user pays rounds up.

9.2 Tokens in and out

Real wNEAR and USDC (NEP-141 tokens [4]) enter only through `ft_transfer_call`. Its `msg` field names the action, for example `split`, `conditional_sell`, `sell_and_borrow`, `post_order`, `supply`, `repay`, `deposit_collateral`, `liquidate`, `pledge` or `fund_insurance`. There is no separate approval step. The contract returns the part the action did not use, and the token contract refunds it; if any check fails the call panics and the token contract refunds everything. Inside the contract every amount has 18 decimals: USDC (6 decimals) converts exactly; for wNEAR (24 decimals) the part below one internal unit is sent straight back. Payouts use `ft_transfer` with a 5 Tgas callback that credits the amount back if the transfer fails (for example when the receiver is not registered with the token).

9.3 Storage staking

NEAR charges for on-chain storage by locking NEAR [5]. Origami Verse implements NEP-145: an account registers with at least 0.1 NEAR. Every state-changing call measures storage before and after, charges the bytes it adds to the payer's deposit (failing if the deposit does not cover them) and credits the bytes it frees. Zero balances are deleted so they stop costing storage. A new market's storage is paid from its proposer's opening reserve, then from a protocol storage fund. The contract code itself (1,380,136 bytes) locks about 13.8 NEAR at NEAR's price of 10^{19} yoctoNEAR per byte.

9.4 Gas

A NEAR transaction may attach at most 300 Tgas. The web app attaches 150 Tgas to routine token calls (split, sell or buy from the wallet, supply, repay, deposit collateral) and the full 300 Tgas to pledges, liquidations, proposals, openings and resolution requests. World trades from the order screen attach 20 Tgas per order fill, 30 per pool swap and 25 per posted order. A pledge schedules an automatic opening only if 215 Tgas are still left. The OutLayer callback receives 20 Tgas. Testnet buys the attached gas up front and refunds what is not burnt, so an everyday action needs about 0.17 NEAR free while signing but costs about 0.001 NEAR. A reference update by the keeper burnt 3.7 Tgas (0.00035 NEAR).

9.5 Events, views and keys

Every action logs a NEP-297 event [7] under the standard `multiverse`, and every balance change logs NEP-245 `mt_mint`, `mt_burn` or `mt_transfer`. Views are designed one per screen, because NEAR RPC has no batching. The reporter signs with function-call keys limited to `report_outcome`, `finalize` and `report_reference`.

9.6 Upgrades and pausing

The contract carries a code version (2.0.0) and a state version (2). An upgrade deploys the new code and runs `migrate` in the *same* transaction, so a failed migration reverts the deployment as well. Ownership moves in two steps (propose, then accept). The owner, or an optional guardian, can pause narrow parts of the system with separate flags: trading, lending, minting, transfers, liquidation, the oracle, pledging and exits. After an unpause, liquidations can be held back by a grace period of at most 86,400 seconds so borrowers have time to react.

9.7 The keeper

A hosted **keeper** (a serverless function next to the web app) does three jobs in each round: it posts the NEAR/USD reference when the stored one is at least 300 seconds old; it keeps the lendable market close to the reference by trading back towards it when the implied spot is more than 4% away (and leaves gaps over 25% to a person); and it liquidates unhealthy loans, taking cash only when the cash covers the repayment after a 0.5% margin. It runs when anyone has the site open (at most once a minute per browser), on a daily cron and from an external scheduler every 30 minutes. Two rounds cannot double-send: each key signs with a nonce derived from the block height rounded down to 100 blocks, so a second round in the same window is rejected by the chain at no cost.

10 Security and invariants

10.1 What the contract checks

- **Backing** (Equation 2) for every market, asset and possible world, and that the contract's real token balance equals reserves plus users' internal balances.
- **Dead stays dead**: nothing on a dead verse succeeds and its supply never changes.
- **Resolution never lowers a health factor.**
- **Liquidation covers the repayment**: when the worst-world value covers repayment plus bonus, the seized slice is worth at least the repayment in every possible world.

- **Order escrow**: every open bid holds at least the cost of what remains, every open ask exactly what remains.
- **No fill after resolution or expiry**, and only the maker can cancel.

Each is a stateful property test: random sequences of 50 to 200 actions (256 cases each, and 24 to 40 cases on eight-world markets), with the invariant checked after every step. Market-creation invariants (pledges, refunds, openings) have their own randomised tests. The full Rust suite had 856 passing tests at the last recorded integration run.

10.2 Internal review

Before version 2, the team reviewed the first version and recorded ten findings (two high, four medium, four low), each first reproduced by a failing test and kept as a regression test. The two high findings were about lending prices coming from the protocol's own pools, which then used a 60-second TWAP: pumping every world pool for one window could unlock too much borrowing, and dumping one world pool for one window could liquidate a healthy borrower. Both are mitigated, not removed: the 30-minute TWAP, the price states of Section 4, the borrow/liquidation split and, in version 2, the outside reference. The version 2 specification went through an adversarial review of its economics, runtime limits, migration and completeness before it was implemented.

10.3 What is not covered

- There has been **no external audit**. The review above was done by the people who built the system.
- The owner is trusted: `admin_resolve` finalizes at once and only the owner can dispute. There is no timelock and no multisig.
- One reporter key proposes outcomes and posts the reference. If the owner does not dispute a wrong proposal in time, anyone can finalize it.
- Someone who holds a pool at a false price for a whole TWAP window still moves it; on testnet few people arbitrage it back. The outside reference limits but does not remove this.
- wNEAR and USDC on testnet are mock tokens with an open mint; they have no value.

11 Deployment and evaluation on testnet

11.1 Accounts

Origami Verse v2 runs on NEAR testnet. The contract is `app.ifverse-staging.testnet` (code hash `9K62NFX7gQiGajRuLTFcPy7F9n5nADSrFgEStUSFcRR`, deployed in transaction `C3KthRtFPy...`). Mock wNEAR (24 decimals) and USDC (6 decimals) live at `wnear.ifverse-staging.testnet` and `usdc.ifverse-staging.testnet`. The reporter, keeper and maker have their own sub-accounts. The web app is at `https://multiverse-lending.vercel.app`. Transaction hashes below are shortened; each is a link to its full record on `https://testnet.nearblocks.io`.

11.2 Live markets

Table 3 lists the ten markets. Market 1 is the lending market: each of its tiers was supplied with about 100,000 USDC, and at 16:39 UTC on 14 November 2026 each tier had two borrowers (Safe 1,330.56, Normal 2,641.09 and Bold 2,504.25 USDC borrowed). Pledged markets have world

Id	Events	Worlds	Origin	Status	Goal
1	SPICE × Fed December (lending, three tiers)	4	team (anchor)	open	–
2	Fed October (past)	2	team	resolved HOLDS	–
3	NEAR close > $4.00 on Nov 1 (past)	2	team	resolved YES	–
4	Fed December	2	Kind B, proposal 1	open	42,000
5	SPICE	2	Kind B, proposal 2	open	41,000
6	NEAR close > $3.25 on Dec 31	2	Kind A, proposal 3	open	40,000
7	SPICE × NEAR Dec 31	4	Kind B, proposal 4	open	54,000
8	SPICE × Fed Dec × NEAR Dec 31	8	Kind B, proposal 5	open	66,000
9	NEAR mainnet > 30M txs in Dec 2026	2	Kind A, proposal 14	open	39,000
10	Fed December × NEAR Dec 31	4	Kind B, proposal 15	open	55,000

Table 3: Markets on `app.ifverse-staging.testnet` (`list_markets`, 14 Nov 2026). Goals are in USDC. Seven of the ten markets were opened by proposals and pledges, not by the team. By 16:40 UTC on 14 November, 21 proposals had been made.

pools of roughly 15,000–20,000 USDC each, below the 50,000 USDC depth lending requires, so they are trading-only until they deepen.

11.3 Real transactions

All flows were exercised with real testnet transactions. A selection:

- Split, sell and buy on market 1: `DadzKfXqrV...`, `FJf2F6YKWU...`, `C5hCvxEWDF...`.
- A four-world hedge in one call (sell in three worlds, keep the fourth): `8tBtiLdLGc...`.
- Order place, cancel and fill: `8vCsyVaizg...`, `2WaRDhGcT3...`, `BoDLFFgRqF...`.
- Borrow in the Bold tier: `F41xJKvnvH...`.
- A Kind A proposal and its approval, then a Kind B proposal and the pledge that opened its market: `39Eu1oWYGZ...`, `7etK5dUxeD...`, `F69pf735E5...`, `5vFZLCzAni...`.
- A Kind A proposal recorded for the walkthrough video (it became market 9): `91RJbDnwkx...`.
- Reference update by the keeper: `HTYgfTkHfx...` (3.33245, from four sources with a 114 bps spread).

11.4 A real liquidation

To show liquidation end to end, one account sold 17,600 wNEAR-if-DELAYED·HOLDS into that world's pool, moving NEAR's price in DELAYED·HOLDS from about 2.61 to 1.58 USDC. As the 30-minute average followed, a Bold-tier borrower's health factor fell from 1.21 to 0.98 in about 12 minutes. One keeper round found the loan and liquidated it (`4p5K22Nqtu...`). It chose tokens rather than cash, because selling the seized wNEAR into the pushed pool would not have covered the repayment, exactly the rule the keeper follows. The price was then bought back.

11.5 Keeper behaviour

In testing, two keeper rounds raced in the same 100-block window: one landed (`FPmqoG6kh4...`) and the other was rejected for its nonce without reaching the chain, as designed. In two later recorded rounds the market maker step found market 1 within its 4% band (42 and 10 basis points from the reference), so it sent nothing, and borrowing stayed open.

12 Limitations and future work

- **Testnet only.** Nothing has been deployed or tested on mainnet. Tokens are mock tokens.
- **Trusted roles.** The owner can resolve and dispute; one reporter proposes outcomes and posts the reference. Future work: a timelock, a multisig owner, bonded reporters and open disputes.
- **Liquidity.** Lending reads only Origami Verse's own pools. On testnet they are deep only where the team and pledgers put liquidity, and few people arbitrage them. Only market 1 is lendable today.
- **Odds sources.** The SPICE odds are an assumption and the Fed odds a dated reading; a live outside odds feed would make implied prices more meaningful.
- **Slow price response.** Because lending uses a 30-minute average, a real price move reaches borrowing limits and liquidations over up to 30 minutes.
- **Events.** A market combines at most three binary events (eight worlds). Events with more than two outcomes are not supported.
- **Reference path.** The NEAR/USD reference is written by the reporter on testnet; the pinned TEE program for it is built but not yet configured on the live contract.
- **Audit.** An external security audit is required before any real value is involved.

13 Conclusion

Origami Verse turns "what if" into something you can hold. A real asset splits into one token per possible world, each world gets its own price, and a loan is judged by the worst world that can still happen, so it stays covered whatever the news. Around this core sit dynamic-fee pools, an order book, three lending tiers with insurance, markets that anyone can propose and fund, resolution from public data with a dispute window and a TEE option, and an outside price check. The whole system is one NEAR contract, live on testnet, with every number in this paper readable from its code and views.

References

[1] D. White. *Multiverse Finance*. Paradigm, May 2025. https://www.paradigm.xyz/2025/05/multiverse-finance

[2] NEAR Protocol documentation. https://docs.near.org

[3] near-sdk-rs, the Rust SDK for NEAR smart contracts. https://github.com/near/near-sdk-rs

[4] NEP-141: Fungible Token Standard. https://github.com/near/NEPs/blob/master/neps/nep-0141.md

[5] NEP-145: Storage Management. https://github.com/near/NEPs/blob/master/neps/nep-0145.md

[6] NEP-245: Multi Token Standard. https://github.com/near/NEPs/blob/master/neps/nep-0245.md

[7] NEP-297: Events. https://github.com/near/NEPs/blob/master/neps/nep-0297.md

[8] NEP-519: Yield Execution. https://github.com/near/NEPs/blob/master/neps/nep-0519.md

[9] OutLayer: verifiable off-chain execution for NEAR (contract `outlayer.testnet`). https://app.outlayer.ai

[10] H. Adams, N. Zinsmeister, M. Salem, R. Keefer, D. Robinson. *Uniswap v3 Core*. 2021. https://uniswap.org/whitepaper-v3.pdf

[11] Uniswap v4 core, `TickMath`. https://github.com/Uniswap/v4-core

[12] Federal Reserve Bank of St. Louis, FRED series DFEDTARU and DFEDTARL. https://fred.stlouisfed.org/series/DFEDTARU