Lora Finance: Renting Upside

Streaming, call-like convexity exposure with bounded downside and no price-based liquidations

lora.finance

Abstract

Lora Finance is a MegaETH-native DeFi protocol that provides an options-like upside product via streaming payments. Instead of borrowing (and risking liquidation), users pay a continuous per-second fee to maintain upside exposure to an underlying asset. Liquidity providers deposit the underlying into an ERC-4626 vault; the protocol allocates notional exposure from vault inventory, sets a strike from an oracle price (optionally out-of-the-money via a user-selected delta), and settles a capped call-like payoff if the underlying finishes above strike.

The design goal is to deliver capital-efficient exposure with transparent costs and a "service termination" failure mode (stream stops) rather than price-triggered liquidations.

1. Problem and Motivation

The October 10, 2026 liquidation cascade - during which more than $21 billion in perpetual futures positions were forcibly closed within minutes - exposed a structural weakness in crypto's dominant derivative instrument. Both retail and institutional participants saw auto-deleveraging engines close positions at severely adverse prices during brief volatility spikes, not because market beliefs changed, but solely due to protecting platform solvency. Positions were terminated by system design, not trader intent.

This event crystallized a broader realization: perpetual futures fail to preserve directional exposure under conditions of elevated volatility. In traditional financial markets, options volumes far exceed futures volumes precisely because they offer parametric risk control. Traders can define maximum loss, face no margin calls, and are never forcibly exited due to transient price movements. The October 2026 events demonstrate that crypto markets have reached a maturity level where these same properties are no longer optional - they are required.

Despite this need, existing decentralized options protocols face substantial adoption barriers. Conventional options demand familiarity with implied volatility, Greeks (delta, gamma, theta, vega), and expiry management, imposing significant cognitive overhead that

restricts participation to specialists. Fixed expiries additionally require traders to predict timing as well as direction, manage rollovers, and accept that being "right but early" results in total loss. Liquidity further fragments across numerous strike–expiry pairs, reducing effective depth and increasing slippage. This design paradigm is poorly aligned with crypto's 24/7, narrative-driven trading environment, where participants require flexible entry and exit without committing to specific time horizons. As a result, a persistent product-market gap remains between what traders need - liquidation-free directional exposure with convex payoff profiles - and what existing options infrastructure provides: complexity, expiry risk, and fragmented liquidity.

This paper introduces Lora Finance, which reconceptualizes options as a streaming upside rental mechanism. Instead of paying an upfront premium for a fixed expiry, users pay a continuous, per-second fee to maintain upside exposure for as long as they choose. This structure eliminates expiries, premium estimation, and Greek-based risk management, while preserving convex payoff characteristics. By transforming options from discrete contracts into continuous exposure, Lora makes options-like risk profiles accessible across the market spectrum - from retail traders seeking leveraged upside without liquidation risk, to institutions requiring hedges that cannot be force-closed during periods of market stress.

The objective is to establish everlasting options as the natural successor to perpetual futures, delivering directional exposure and convexity with the reliability and risk control that post-October-2026 crypto markets demand.

2. Protocol Overview

2.1 Renting Upside (intuition)

When a user opens a Lora position, they select an asset and the exposure they want (e.g., "upside of 1 ETH"), and begin a Superfluid stream that continuously pays rent. While the stream is active, the position's PnL tracks the underlying's price movement, similar to holding that notional amount. If price moves against the user, the position shows a drawdown but is not liquidated solely due to price—the position remains open as long as the rent stream continues.

2.2 High-level mechanism

At a high level:

1. LPs deposit the underlying asset into a vault (ERC-4626).
2. A user opens (or increases) a Superfluid stream into the market.
3. Using current utilization, the protocol computes a rent rate and maps the user's spend-per-second into a maximum notional exposure.

4. A strike is set from an oracle spot price, optionally shifted out-of-the-money using a user-selected delta.
5. On close, the protocol settles a capped call-like payoff from the vault if the position is in-the-money.

2.3 Key properties

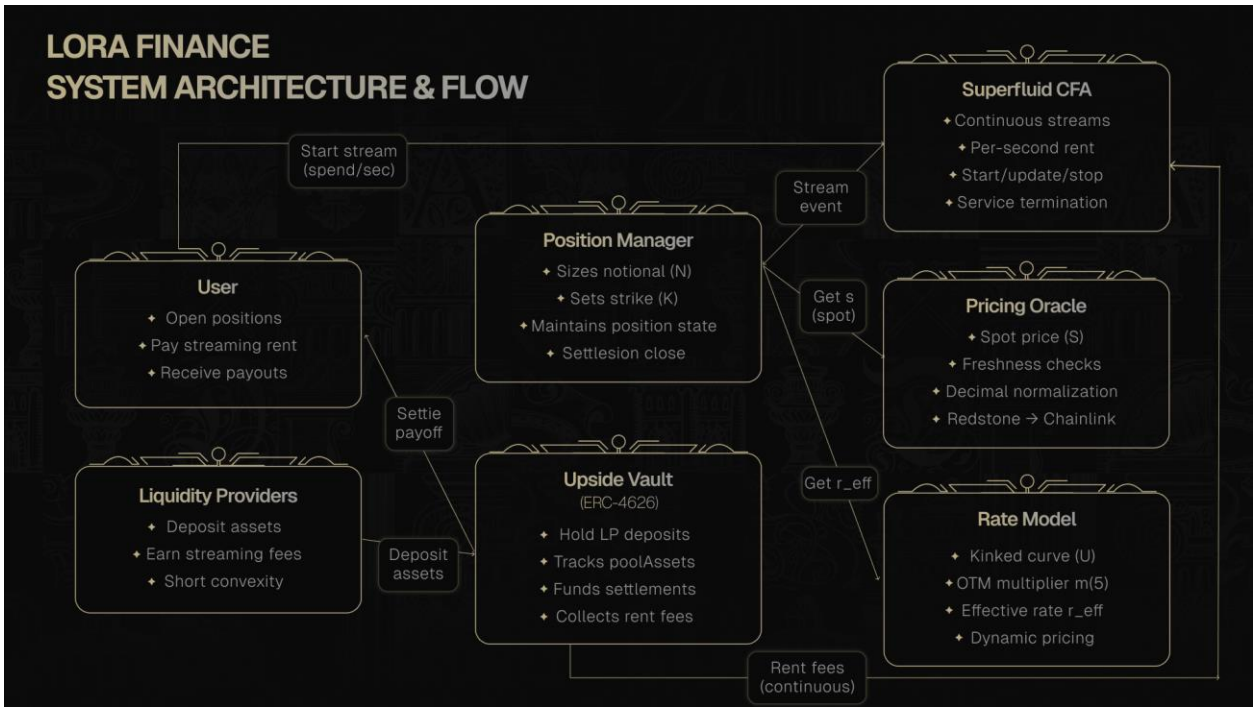
- Capital efficiency: users can access exposure without buying the full underlying or posting large collateral.
- No price-based liquidation: volatility does not trigger forced closes; the stream acts as the payment primitive.
- Continuous duration: positions have no expiry—users pay only for the time they keep exposure.
- On-chain and permissionless: settlement and accounting are executed by smart contracts.
- Low friction UX on MegaETH: fast, low-cost transactions make position adjustments feasible.

3 System Architecture

3.1 Core components

Lora can be implemented as an ERC-4626 “Upside Vault” integrated with Superfluid Constant Flow Agreements (CFA). A reference architecture consists of:

- **Upside Vault (ERC-4626)**: holds LP deposits, tracks assets, and funds settlements.
- **Market / Position Manager**: sizes notional, sets strike, maintains position state, and settles on close.
- **Superfluid payment stream:** continuous rent charged per second; streams can be started, updated, or stopped.
- **Pricing oracle wrapper:** normalizes decimals and enforces freshness checks (e.g., Redstone initially, migrating to Chainlink).
- **Rate model:** maps utilization to a per-second rent rate (kinked “jump” curve).



3.2 Settlement and service termination

If a user's stream stops (e.g., the user removes the stream or their balance is insufficient), the protocol closes the position as a service termination event. This is distinct from margin liquidation: the close is driven by payment failure rather than a collateral threshold.

4 Core Mathematical Specification

4.1 Pool utilization

Let poolAssets be total vault assets (underlying units) and lockedNotional be the sum of notionals across open positions. Utilization U is defined as:

$$U - Utilization$$

$$U = \begin{cases} 0 & \text{if poolAssets} = 0 \\ \dfrac{\text{lockedNotional}}{\text{poolAssets}}, & \text{otherwise} \end{cases}$$

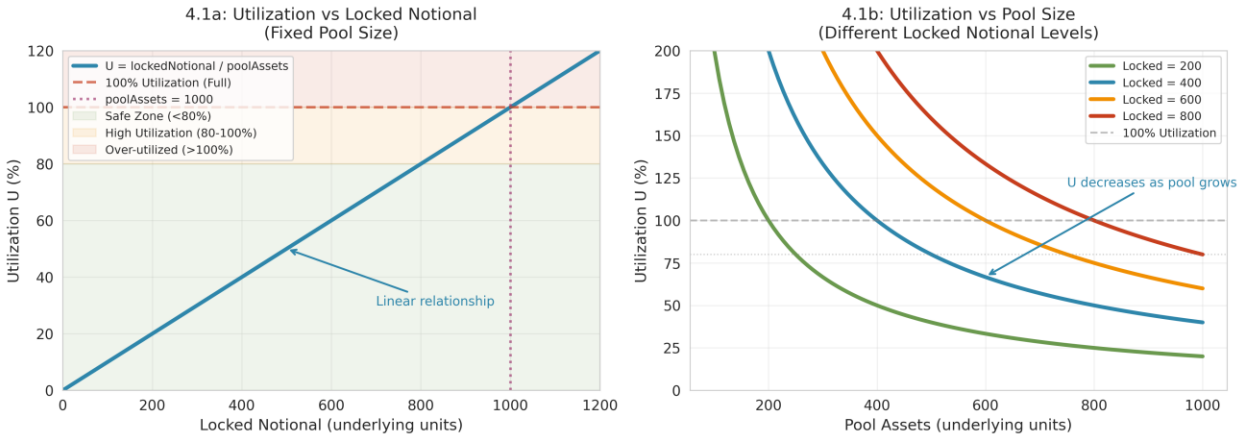


Figure 4.1: Pool Utilization Dynamics

Panel (a) illustrates the linear relationship between locked notional and utilization for a fixed pool size of 1,000 units. Panel (b) demonstrates how pool size affects utilization for fixed amounts of locked notional.

Key insights on pool utilization: (1) Utilization is a fundamental control variable that drives the rent rate through the kinked curve in Section 4.2; (2) The protocol must constrain locked notional to prevent over-utilization (U > 100%), which would imply insufficient backing for open positions; (3) LP incentives must be sufficient to attract capital that keeps utilization in the safe operating range, ensuring both competitive rent rates for users and adequate liquidity buffers for settlements.

4.2 Jump (kinked) rent curve

A kinked curve raises rent as utilization increases.

r_{base} - Rate per second

$\tilde{r}(U)$ denote the raw (unclamped) per-second rate from the kinked curve.

s_1 - Slope before kink

s_2 - Slope after kink

κ - Kink

clamps: r_{min}, r_{max}

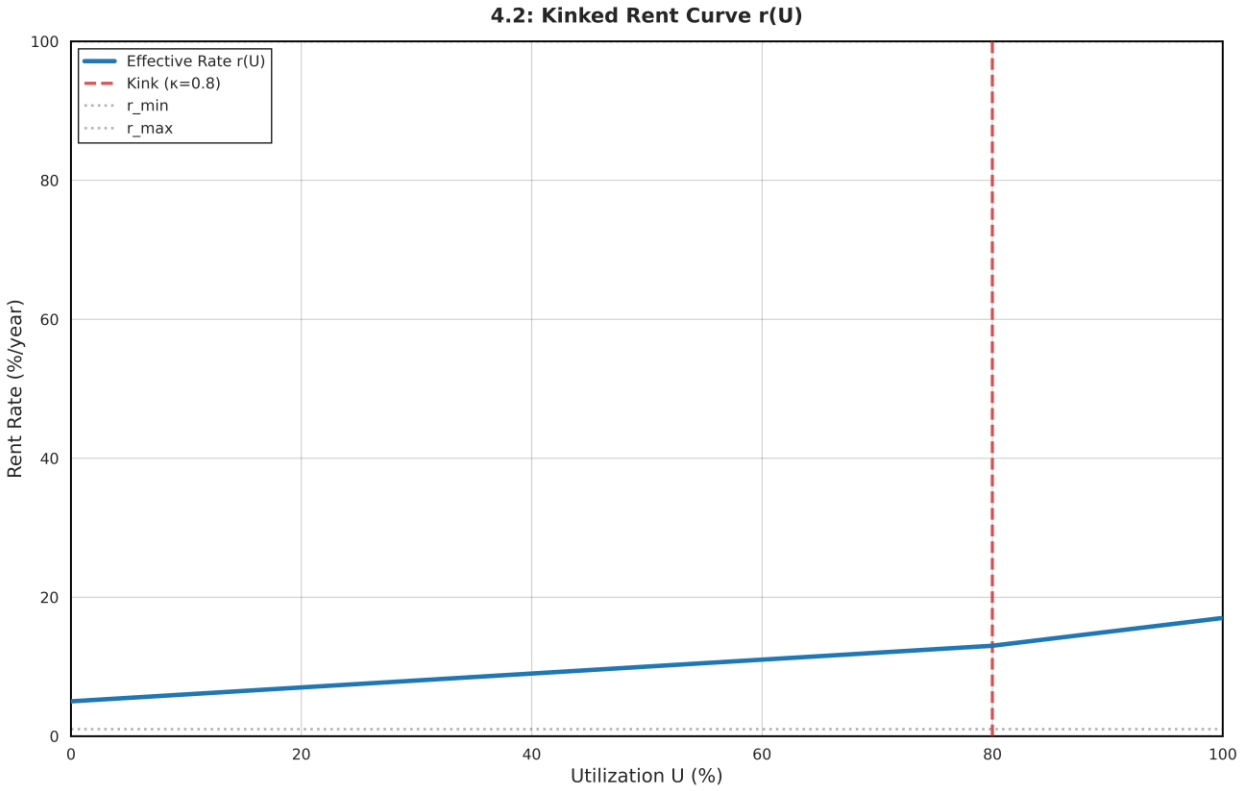
Domain: $U \in [0,1]$ is utilization (WAD fraction).

$$\tilde{r}(U) = \begin{cases} r_{\text{base}} + U\, s_1, & U < \kappa \\ r_{\text{base}} + \kappa s_1 + (U - \kappa) s_2, & U \geq \kappa \end{cases}$$

Then clamps,

$$r(U) = \min(\max(\tilde{r}(U), r_{\min}), r_{\max})$$

The kinked rent curve shows how rent rates increase with utilization, featuring two distinct regimes separated by a "kink" at κ (typically 80%).



Lora kinked rent curve implies two-regime pricing:

- Below kink (U < 80%): Gentle slope encourages capital-efficient usage at competitive rates
- Above kink (U ≥ 80%): Steep slope discourages over-utilization and protects vault solvency

These are some practical examples with shown parameters:

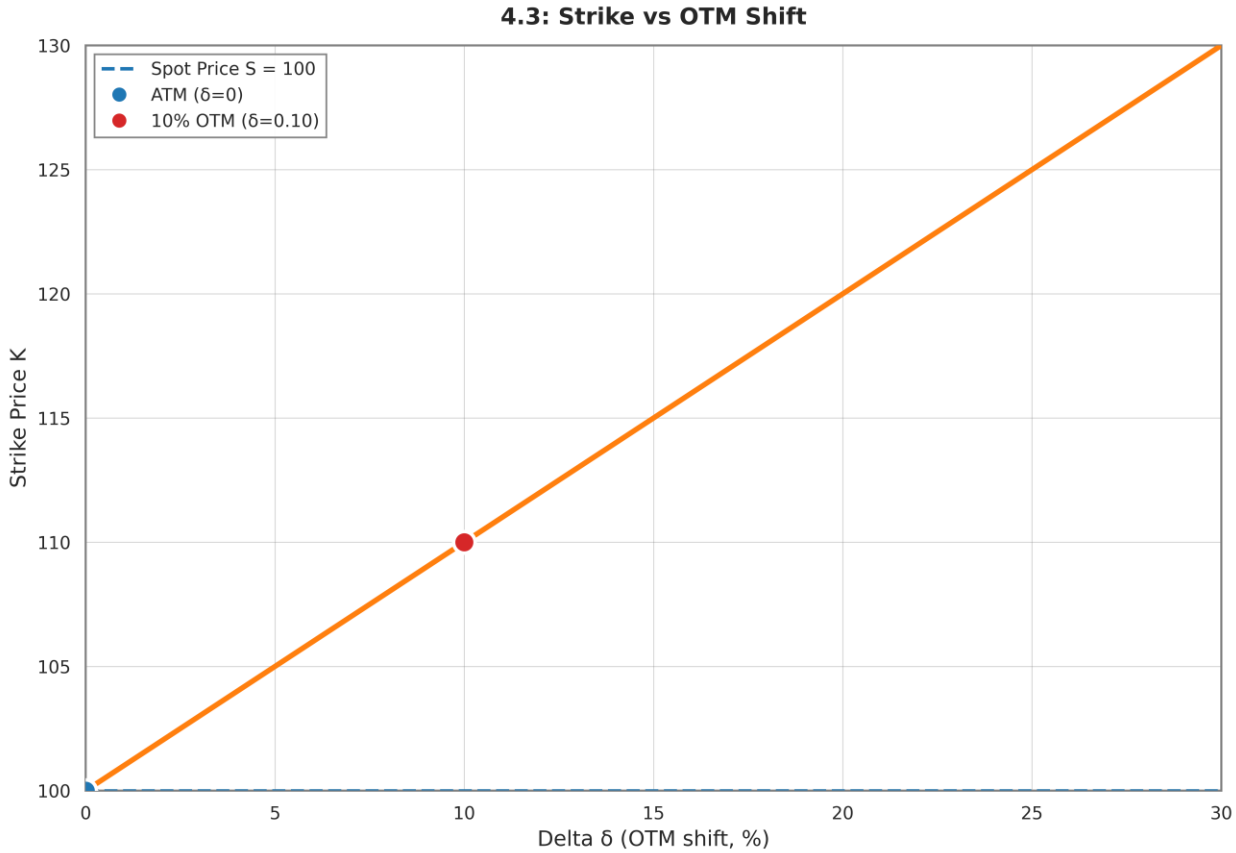
- At U=30%: ~4.4% annual rent (competitive)
- At U=70%: ~7.6% annual rent (normal)
- At U=90%: ~17% annual rent (expensive, pressure to reduce)

4.3 Strike and delta (OTM shift)

S - Spot (comes from oracle, 18 decimals)
δ - Strike delta (WAD fraction) >=0
K - Strike (18 decimals)

$$K = \begin{cases} S \cdot (1+\delta), & \delta > 0 \\ S, & \delta = 0 \end{cases}$$

The following chart shows the linear relationship between OTM shift δ and strike price K, with the formula: K = S × (1 + δ).



The tradeoff for renters are illustrated as following:

- ATM (δ = 0%): Strike = Spot → Immediate exposure, highest rent
- OTM (δ > 0%): Strike > Spot → Delayed exposure, discounted rent

4.4 OTM multiplier (rate discount)

To discount rent for out-of-the-money strikes, define a monotone decreasing multiplier m(δ) controlled by parameter h (OTM half):

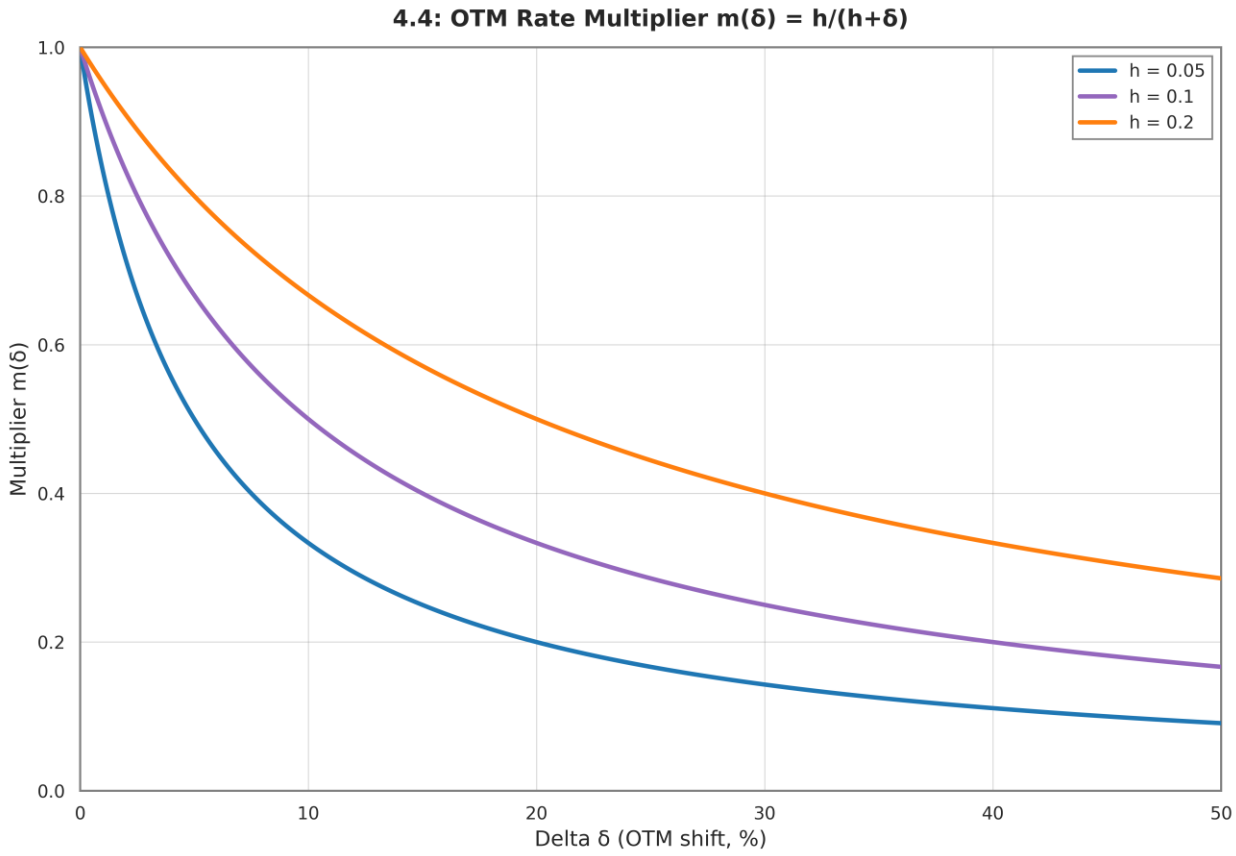
h - OTM half, scales the effective rate when the trader chooses $\delta > 0$
m - Rate multiplier

$$m(\delta) = \begin{cases} 1, & h + \delta = 0 \\ \dfrac{h}{h+\delta}, & \text{otherwise} \end{cases}$$

The effective per-second rate charged to size notional becomes:

$$r_{\text{eff}}(U, \delta) = r(U) \cdot m(\delta) = r(U) \cdot \frac{h}{h+\delta}\,.$$

The following chart illustrates how the multiplier m(δ) = $\frac{h}{h+\delta}$ decreases as positions move further out-of-the-money, providing rent discounts. Three curves show different discount aggressiveness based on parameter h.



The discount formula works as following:

- At δ = 0% (ATM): m = 1.0 → Full rent (no discount);
- At δ = h: m = 0.5 → 50% discount;
- As δ increases: m approaches 0 → Asymptotic discount;

where effective rate: r_eff(U,δ) = r(U) × m(δ)

4.5 Notional sizing from spend-per-second

Let N be notional exposure (underlying units) and $spend_{max}$ be the user's Superfluid flow rate (tokens/second). The instantaneous spend for notional N is:

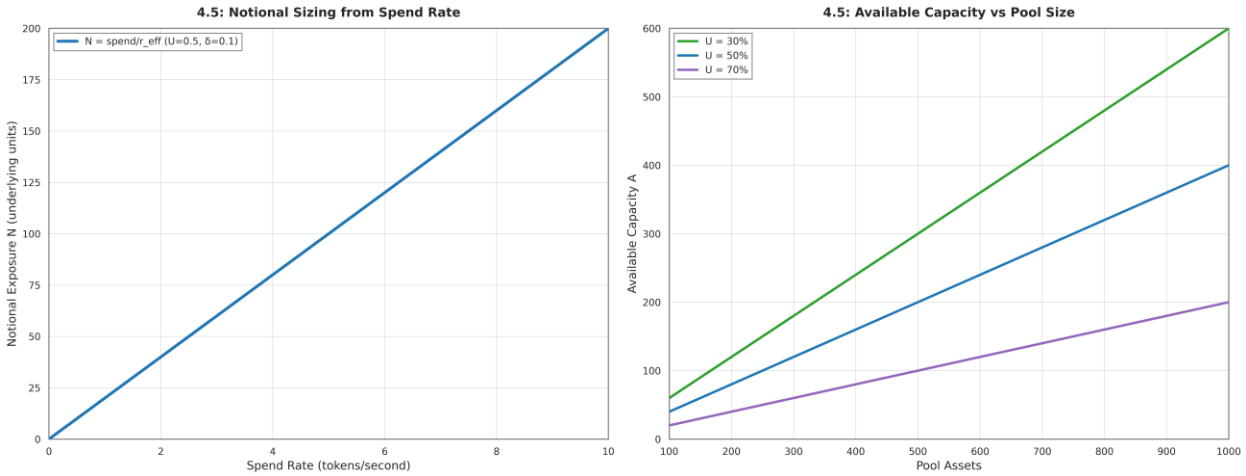
$$spend(N; U, \delta) = N.r_{eff}(U, \delta)$$

The protocol chooses the largest N such that spend(N;U,δ) ≤ $spend_{max}$, subject to inventory constraints and discrete lot sizing.

Additionally, inventory/cap constraints are enforced in discrete lots:

b - Buffer fraction (WAD)
B - Buffer amount
A - Available asset cap = *max(0, poolAssets−lockedNotional−B)*
L - Lot size
M - Max lots = *[A/L]*

The solver binary-searches *m*∈[0,*M*] to maximize *N*=*m*·*L* under the spend constraint.



The left chart shows notional sizing from spend rate, with the linear relationship

N = $\frac{spend_{max}}{r_{eff}}$ between a user's continuous payment rate and the notional exposure they receive. Users control their Superfluid stream flow rate, and the protocol automatically sizes their position proportionally. Higher spend rates grant larger notional exposure, with the exact ratio determined by the effective rent rate $r_{eff(U,\delta)}$.

The right chart illustrates the available capacity vs the pool size, demonstrates how available capacity A = poolAssets - lockedNotional - B scales with pool size at different utilization levels. A 10% buffer (B) is always reserved for solvency. As the pool grows through LP

deposits, available capacity increases linearly, but at rates dependent on current utilization—creating natural supply/demand balance through the rent curve mechanism.

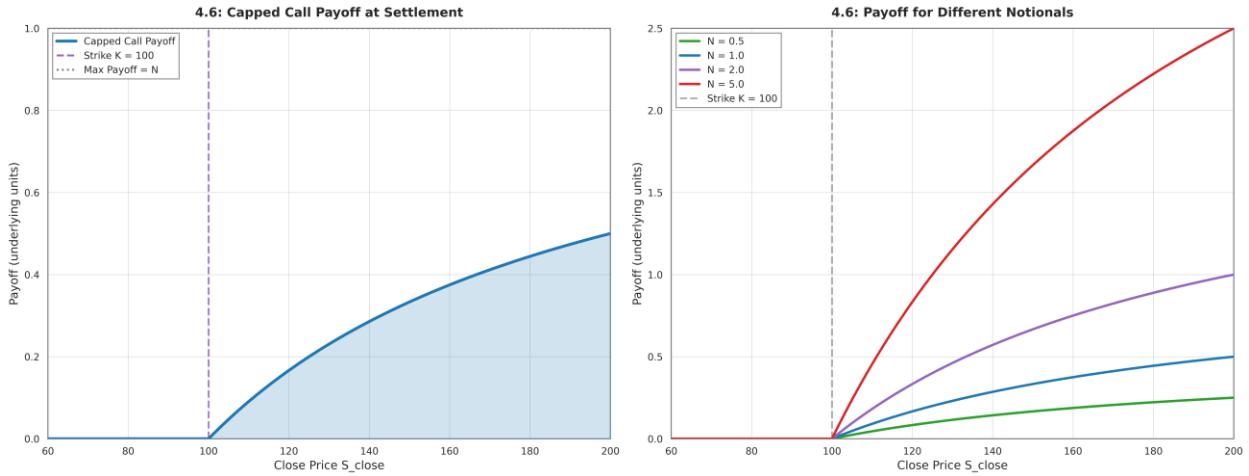
The left chart shows what users can request (notional from spend), while the right chart shows what the protocol can provide (capacity from pool). When capacity is constrained, rent rates rise (Section 4.2), which both reduces demand (left chart shows less N per spend) and attracts LP supply (right chart shows pool growth).

4.6 Payoff at close (capped call-like)

On close, with final oracle price S_{close} the payoff in underlying units is capped by N:

$$payout\ (N,\ K,\ S_{close}) = N.\max\left(0,1 - \frac{K}{S_{close}}\right)$$

This payoff is zero when S_{close} ≤ K and approaches N as S_{close} → ∞.

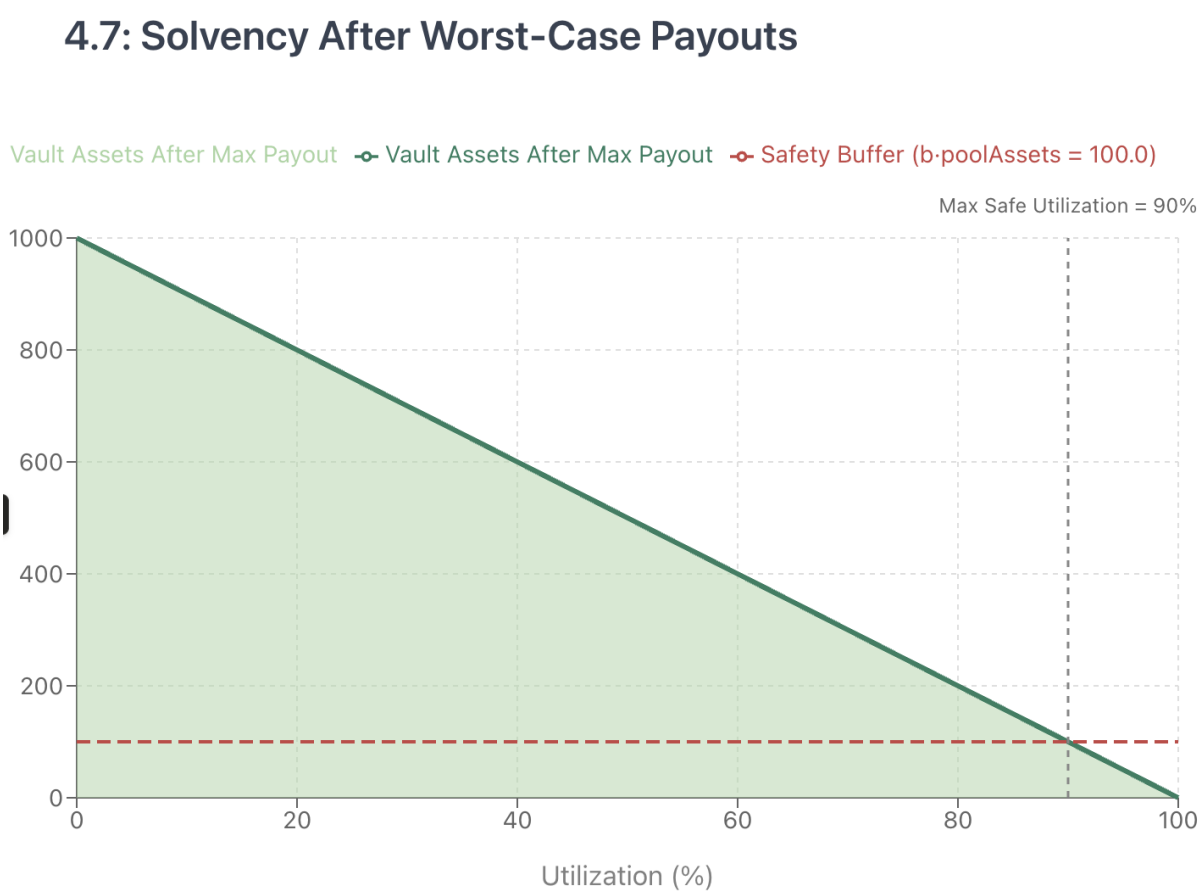


The left chart shows the payoff function payout = N × max(0, 1 - K/S_close) for a position with notional N=1 and strike K=100, while the right one illustrates how payoff scales linearly with notional size, with strike K from Section 4.3, notional N from Section 4.5, cap ensures solvency in Section 4.7. User's net PnL is calculated by the formula: PnL = payout - total_rent_paid.

4.7 A solvency sketch

For any position i,$payout_i \leq N_i$. Therefore $\Sigma payout_i \leq \Sigma N_i$ = lockedNotional.

If risk parameters ensure lockedNotional ≤ (1 - b)*poolAssets, then even in the worst-case where all positions close deep in-the-money—the vault retains at least b*poolAssets after paying maximum payouts.



5 Position Lifecycle and Examples

5.1 Open → maintain → close

A simplified lifecycle is:

- Open: user starts or increases a stream; protocol sizes notional, sets strike, records the position.
- Maintain: PnL updates with price while rent accrues continuously; user may adjust exposure by updating the stream.
- Close: user stops or reduces the stream; protocol settles payoff using a fresh oracle price and unlocks notional.

5.2 User Flow: Leveraged Upside via Streaming

Consider a user with limited capital who wants 2× exposure to an asset. They deposit funds to cover streaming payments and open a position whose notional is sized from their chosen spend-per-second. If the asset rises, the user receives profits as if they held the larger

notional; if the asset falls, the user sees a drawdown but is not liquidated due to price. They can close at any time to stop rent accrual.

5.3 LP Economics: Fee Yield and Convexity Risk

An LP deposits the underlying asset into the vault. When users rent upside, the vault's inventory backs their notionals and accrues streaming fees. In return for earning rent, the LP is implicitly short convexity: if the asset rallies strongly, some upside is paid out to renters at close.

5.4 Composability: Stream-Based Position Integrations

Because positions are on-chain and parameterized by streams, external apps (wallets, portfolio managers, or DeFi protocols) can integrate Lora to offer "boost exposure" or hedging-like primitives without implementing liquidation engines or perp margin logic.

6. Competitive Landscape

Lora's payoff resembles a perpetual call-like position, but its pricing and risk mechanics differ from common alternatives:

- **Borrowing / margin lending:** often over-collateralized and liquidation-prone; Lora has no debt position and no price-based liquidation.
- **Perpetual futures:** require margin and can liquidate; funding can be volatile; Lora charges a one-way, algorithmic rent stream with vault-backed settlement.
- **Traditional options:** have expiry and premiums; Lora is "pay-as-you-go" with no expiry as long as the stream continues.
- **Leveraged tokens:** avoid liquidation but suffer volatility decay; Lora charges explicit rent with no rebalancing-driven decay.

7. Risks and Security Considerations

Lora aims to make risks explicit and to mitigate them via conservative parameters and contract design. Key risks include:

Risk	Typical mitigations
Smart contract risk	Audits, bug bounties, formal testing, incremental rollouts.
Oracle risk	Freshness checks, reputable feeds (e.g., Chainlink), circuit breakers / pausing on anomalies.
Liquidity risk	Utilization-based rent curve, exposure caps, LP incentives, reserves/insurance modules (optional).

Extreme volatility	Buffer b, rate clamps, safety valves during early stages; governance-driven parameter updates.
User error	Clear UI on cumulative fees, alerts, optional stop-conditions and notifications.
Regulatory & market risk	Frontend controls where appropriate; legal review; adaptable parameters and governance.

8. Future Work

Planned and possible extensions include expanding decentralized option exchange; multi-asset markets; richer rate surfaces that incorporate volatility or term structure; improved close primitives and UX safeguards; and governance frameworks for transparent parameter evolution.

9. References

[1] ERC-4626: Tokenized Vault Standard (EIP-4626).

[2] Superfluid Constant Flow Agreement (CFA) and SuperApp architecture.

Appendix A: Variable Definitions (Core Math)

The following table summarizes notation used in Section 4.

Symbol / Name	Meaning	Units
poolAssets	Total vault assets	Underlying tokens
lockedNotional	Sum of open notionals	Underlying tokens
U	Utilization (lockedNotional / poolAssets)	WAD
r_{base}	Base rent rate	1/sec (WAD)
s_1,s_2	Slopes (pre-/post-kink)	1/sec (WAD)
κ	Kink point	WAD
r_{min}, r_{max}	Rate clamps	1/sec (WAD)
S, S_{close}	Oracle spot price	18-dec price

δ	Strike delta (OTM shift)	WAD
K	Strike	18-dec price
h	OTM half parameter	WAD
m(δ)	OTM rate multiplier	WAD
r_{eff}	Effective rent rate	1/sec (WAD)
$spend_{max}$	Stream flow rate	Tokens/sec
N	Notional exposure	Underlying tokens
b	Buffer fraction	WAD
B	Buffer amount	Underlying tokens
A	Available capacity	Underlying tokens
L	Lot size	Underlying tokens