Entangle Labs

# Universal Data Feeds

Whitepaper

Date of publication: 5th March 2025

## Abstract

Universal Data Feeds (UDF) is a next-generation, high-performance oracle network delivering hyper-fast, cost-efficient, and highly secure data feeds across multiple blockchains. Designed for unmatched speed (200ms frequency) and scalability, UDF leverages the Entangle Interoperable Blockchain (EIB) and Layer 2 (L2) infrastructures to provide instant, cryptographically verified data with subsecond latency.

With Byzantine fault-tolerant aggregation, secure finalization, dynamic cost-sharing models, and trustless on-chain consumption, UDF enables real-time, high-frequency data delivery for DeFi, AI-driven applications, and Real-World Assets (RWAs). Supporting both push and pull models, UDF ensures trust-minimized, low-latency updates while optimizing costs for decentralized applications.

This whitepaper details UDF's cutting-edge architecture and explores how it achieves industry-leading efficiency, unparalleled security, and lightning-fast data propagation to power the next generation of decentralized applications.

# Table of Contents

# 1. Introduction

The exponential growth of decentralized finance (DeFi), Real-World Assets (RWAs), AI, and blockchain gaming [1] has driven a surging demand for reliable, high-speed oracles. Oracles feed external (off-chain) data, such as asset prices, interest rates, and real-world events, into smart contracts, enabling a vast range of blockchain applications. However, existing oracles often struggle with the blockchain trilemma of scalability, decentralization, and security, making it difficult to balance cost, speed, and security. Additionally, cross-chain compatibility [2] remains a major challenge for oracle solutions.

UDF solves the trilemma by offloading aggregation to Layer 2 (L2) while maintaining Byzantine Fault-Tolerant (BFT) security. It addresses key challenges through efficient smart contract design, omnichain interactions, horizontal scaling of its Oracle Chain via L2 solutions [3], and a robust security framework.

UDF's smart contract architecture is built for cost efficiency, optimizing oracle updates by minimizing redundant on-chain transactions. This ensures data feeds remain economically viable while maintaining high availability. For optimized speed, the Oracle Chain serves as the core processing hub, offloading data collection and computation to L2 auxiliary blockchains. This enables fast, distributed data aggregation, parallel processing, and real-time updates. The Aggregation Module consolidates multiple data inputs, while the Attestation Module generates on-chain verifiable proofs, ensuring high-speed data delivery without compromising accuracy.

Security and data integrity are reinforced through cryptographic verification mechanisms, with ECDSA signatures [4] ensuring data remains tamper-proof throughout its lifecycle. The Oracle Chain further enhances security by providing a dedicated validation and attestation environment, enabling reliable cross-chain deployments without data corruption risks.

Additionally, UDF supports both push and pull data-delivery models, allowing dApps to optimize performance and costs based on their specific needs.

Through cost-efficient design, scalable L2 processing, and robust security measures, UDF delivers high-performance oracle solutions, ensuring seamless and reliable data feeds for the next generation of Web3 applications.

Existing oracle networks often struggle to balance decentralization, efficiency, cost-effectiveness, modularity, and cross-chain adaptability. UDF differentiates itself through:

- **Flexible Aggregation:** Developers can define aggregation logic (VWAP, TWAP, or median-based).
- **Cost Optimization:** Both Push and Pull models provide efficient pricing structures.
- **Multi-Chain Support:** Unlike most oracles limited to EVM chains, UDF extends to both EVM and non-EVM networks.
- **Scalability via L2 Rollups:** Reduces congestion and ensures real-time data availability.

These innovations make UDF a next-generation oracle solution suited for high-performance dApps.

## 2. Protocol Overview

### 2.1 Terminology

**UDF Oracle:** The core system for data aggregation, signature verification, and on-chain publication.
**Pull Model:** A data-verification method that happens in each user transaction by incorporating oracle votes directly in the transaction's calldata.
**Push Model:** A data publication method where nodes periodically push updated values on-chain.
**Transmitters:** Decentralized agents that sign and attest to data, forming a crucial security layer for UDF's aggregated values.
**EIB (Entangle Interoperable Blockchain):** A specialized oracle chain that coordinates data availability, finalization, and bridging logic.
**L2:** Layer 2 rollups that further boost throughput and reduce latency of the EIB.
**Finalized Snapshot:** A canonical data set (e.g., median prices) aggregated from various sources and attested by the transmitters, to be consumed by dApps.

**VerificationLib:** A smart contract library that validates oracle votes for secure data feeds on target chains.

### 2.2 Key Components

**Data Source Layer:** Collects data from off-chain APIs, centralized exchanges, or blockchains.
**Aggregation Layer:** UDF's decentralized agent network merges multiple data points into an aggregated feed or median-based consensus.
**Finalization Layer (EIB & L2):** Ensures that the aggregated feed is validated, time-stamped, and anchored in a trust-minimized manner.
**On-Chain Oracle Contracts:** Offer read or verification functionalities for dApps, with separate modules for push vs. pull updates.

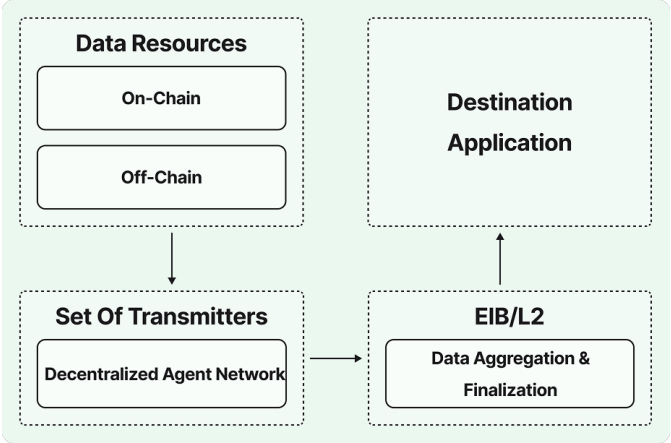


*Figure 1: UDF Architecture Overview: Data Flow from Sources to On-Chain Contracts*

## 3. Architecture

### 3.1 Entangle Interoperable Blockchain (EIB) & L2

UDF leverages the same key properties of the UIP protocol such as security, scalability, decentralization, customizability, and interoperability [5].

UDF leverages the EIB, a specialized blockchain built on Cosmos/Ethermint [6], for orchestrating:

• **Cross-Chain Consensus:** Relaying aggregated data snapshots to each target chain.
• **Validator Set & Finality:** EIB runs a robust consensus mechanism (Tendermint BFT) that finalizes data more rapidly than many L1 networks, enabling near real-time data updates.
• **Layer 2 (L2) Parallelization:** Additional L2 networks parallelized to the EIB to handle specific tasks (aggregator rollups) to offload mainchain overhead and enable unlimited horizontal scaling.

Layer 2 (L2) solutions leverage the Entangle Interoperable Blockchain (EIB) as a trusted layer to periodically commit their state, inheriting EIB's security while achieving greater compute bandwidth and lower costs.

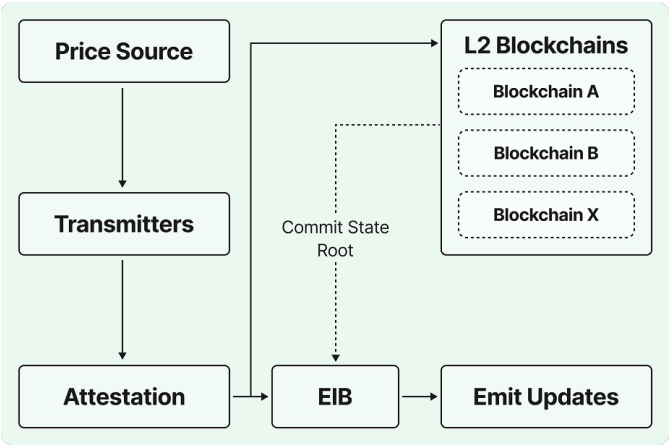


*Figure 2: EIB-L2 Coordination: Secure Data Finality and Scalable Parallel Processing*

**Benefits of EIB + L2**

• **High Throughput:** Capable of batching multiple data updates per block.
• **Instant Finality:** Short block times (~200ms frequency) reduce data latency for both push & pull interactions.
• **Interoperability:** The EIB acts as a "hub" for bridging data across multiple L1 or L2 target chains.

### 3.2 Data Aggregation & Finalization

**1. Raw Data Collection:** Transmitters gather data from on-chain or off-chain APIs.
**2. Aggregation:** Transmitters collectively derive a single consensus using the Volume Weighted Average Price (VWAP) which prioritizes high-volume data to reflect market realities accurately.

$$VWAP = \frac{\sum_{i=0}^{N} P_i \times V_i}{\sum_{i=0}^{N} V_i}$$

Where:

$P_i$ = Price from data source $i$
$V_i$ = Trade volume from data source $i$
$N$ = Total number of sources $i$

This method ensures that high-liquidity markets contribute more heavily to the final oracle value, minimizing volatility manipulation.

**3. Commitment on EIB:** Publishers regularly sign the latest aggregated data and post their vote on the EIB oracle chain.

This architecture not only minimizes the cost on higher-fee blockchains but also ensures that final data is cryptographically proven correct.

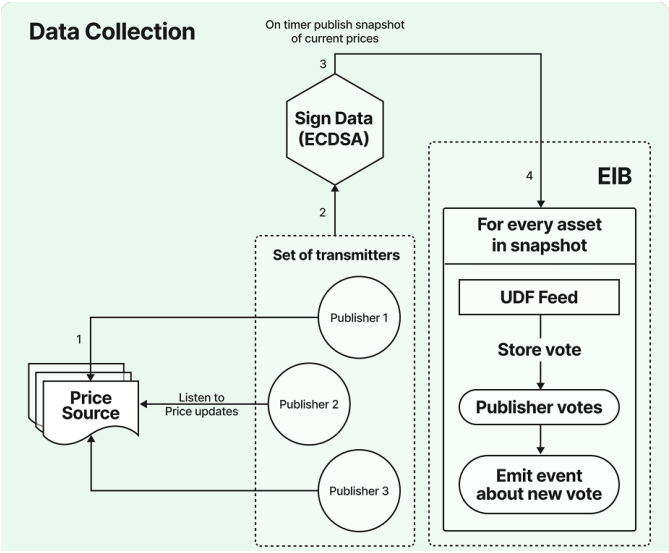


*Figure 3: Data Collection & Finalization: Aggregation with Cryptographic Proof on EIB*

### 3.3 Scalability & Security

**Scalability:** Because data is aggregated off-chain and only a single aggregated result is posted on-chain, UDF avoids exponential overhead. Parallel L2 subnets can handle more specialized or frequent updates, distributing the load based on usage demands.

**Security:** Multiple transmitters must digitally sign each aggregated update. The EIB validator further attests to the correctness of these signatures. If any transmitter is malicious or offline, others can still maintain data continuity, ensuring no single point of failure.

**Byzantine Fault Tolerance & Secure Aggregation**

UDF ensures resilience against malicious actors through Byzantine-tolerant aggregation. Each data update is signed by multiple decentralized transmitters, and a consensus threshold is required before finalization. If any transmitter submits incorrect or manipulated data, the system detects inconsistencies, ensuring data integrity even in the presence of faulty nodes.

The Byzantine Fault Tolerance (BFT) mechanism ensures that at least 50% of transmitters must be honest for accurate aggregation. That means UDF's median-based aggregation can tolerate up to *N/2* faulty transmitters before compromising correctness:

$$f < \frac{N}{2}$$

Where:

$f$ = Number of faulty/malicious transmitters
$N$ = Total number of transmitters

This guarantees a trust-minimized consensus, as long as fewer than half of the sources are compromised.UDF tolerates up to half of transmitters being faulty, ensuring reliability unless a majority colludes.

Furthermore, offline or misbehaving transmitters are penalized, reinforcing honest participation. This model guarantees that data reaching dApps is cryptographically validated and resistant to manipulation.

### 3.4 Role of Transmitters & Decentralized Agent Network

A decentralized network of lightweight transmitter nodes retrieves raw data from various sources and cryptographically signs each update before submission to the oracle chain. These signatures collectively form a Byzantine-tol-

erant proof, ensuring the integrity and validity of the data.

Signatures are merged into a final proof, which is used for validation on external blockchains and off-chain applications.

**Transmitters:**

• **Stake $NTGL tokens for Economic Security:** $NTGL, the native token of the Entangle ecosystem, secures the network via staking. Transmitters must stake $NTGL tokens to participate in the network, aligning incentives with honest operation. Staked assets act as a security deposit, misbehavior, downtime, or incorrect data submissions result in slashing penalties. This mechanism prevents Sybil attacks while ensuring that only committed, well-incentivized nodes contribute to UDF's data integrity.
• **Generate signatures** on data updates, contributing to a trust-minimized consensus.
• **Are monitored and rotated,** with penalties for downtime or producing inconsistent signatures.

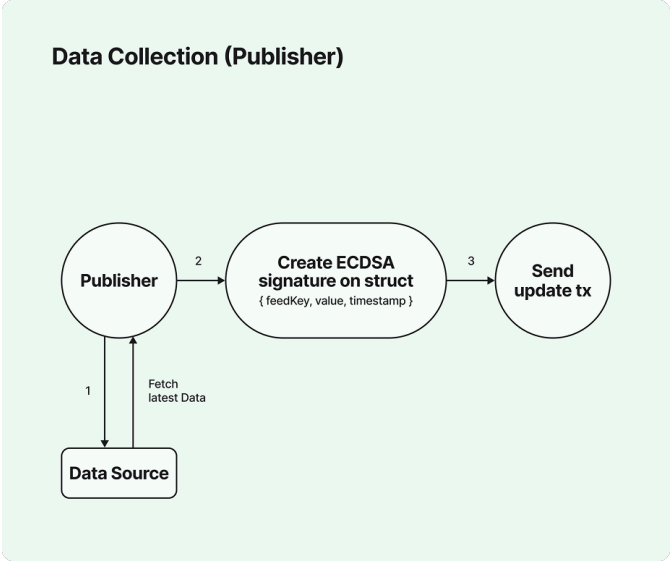


*Figure 4: Transmitter Publisher Flow – Signing Process for Data Consensus Snapshot*

### 3.5 Finalized Data Snapshot

The final snapshot is a cryptographically verifiable object that includes:

• **Aggregated Value** (e.g., BTC/USD = 100,000)
• **Timestamp**
• **Transmitter Signatures**
• **Optional Extra Metadata** (standard deviation, volume data, etc.)

Once finalized, the data snapshot is posted on the EIB/L2 oracle chain, ensuring accuracy through multiple layers of signed votes. The Finalized Data Snap service continuously synchronizes with the oracle chain, providing real-time oracle data for users and push nodes to retrieve validated oracle data efficiently.

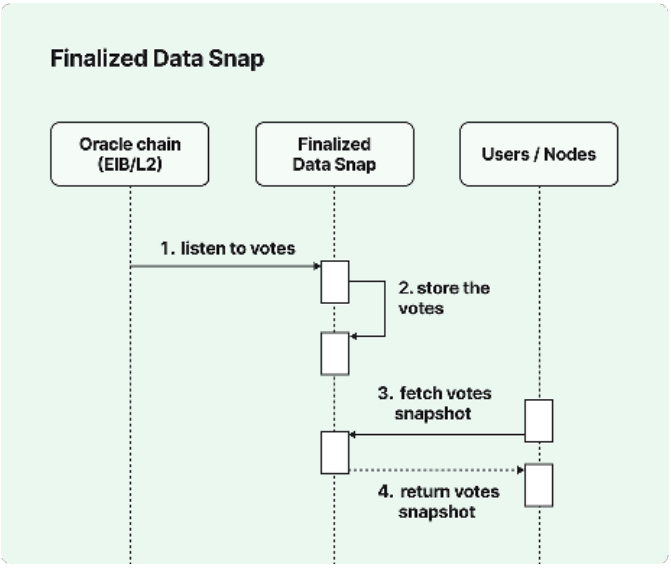


*Figure 5: Finalized Data Snapshot – Validation, Signing & On-Chain Storage*

## 4. Push & Pull Models

UDF provides both push and pull oracle solutions through dedicated marketplace contracts deployed on target chains. These contracts enable seamless data access tailored to different use cases. Pull model is like ordering data on demand; push model is like subscribing to a live feed.

At the core of this system is VerificationLib, a smart contract deployed on target chains to aggregate signed oracle votes into a single, consensus-driven value. It supports multiple data types, including price feeds and high-precision floating-point values, ensuring adaptability across various applications.

Both push and pull marketplace contracts operate along-

side VerificationLib, leveraging its validation mechanism to guarantee the integrity and reliability of oracle data.

### 4.1 Pull Model

**1. User Transaction:** When a dApp user wants to perform an action (e.g., open a leveraged position, or open a lending/borrowing position), the user includes the latest data feed proof in their transaction's calldata.
**2. On-Chain Verification:** The UDF oracle contract verifies signatures from the aggregator (transmitters). If valid, the feed value is accepted for that user's transaction.
**3. Gas & Fee Implication:** Each user pays a "verification fee" in native tokens (either through Pay-As-You-Go (PAYG) approach or predefined monthly subscriptions) to cover the cost of verifying the aggregator's signatures and for the aggregator's services. This suits lower-frequency but latency-sensitive use cases.

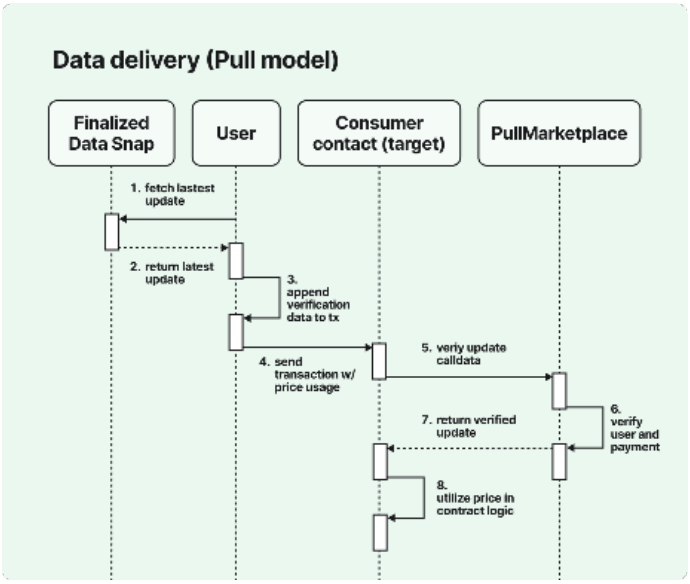


*Figure 6: Pull Model Workflow – User Transaction & On-Chain Verification*

**Advantages & tradeoffs:**

• **Minimal Data Staleness:** The feed is as fresh as the aggregator's most recent update with an update frequency of 200ms.
• **Lower latency:** Data is fetched and verified within the user's transaction, ensuring real-time accuracy.
the user's transaction, ensuring real-time accuracy.
• **Decentralized Access:** Each user includes the proof, removing reliance on a single push node.
• **Fine-Grained Cost:** Perfect for dApps with moderate or unpredictable usage, as they pay only when they need the data.
• **Transaction gas costs:** Each transaction requiring oracle verification incurs verification overhead.
• **Best for on-demand use cases:** Ideal for lending protocols, derivatives, and real-time settlement applications.

### 4.2 Push Model

**1. Scheduled or Triggered Updates:** UDF push nodes post the current aggregated feed to a target chain at given price deviations of 0.25%, 0.5%, 1%, 2.5%, or 5% depending on the deviation parameter selected by users, or based on a predefined heartbeat of 24 hours to guarantee sufficient data freshness even when price variations are minimal.
**2. Single On-Chain Storage:** The UDF push oracle contract holds the verified value so that authorized users' dApps can read it without paying additional verification overhead.
**3. Fee Model:** Subscribed users can share the cost of running the same feed which guarantees cost efficiency for users. UDF charges a small percentage on deposits to the push marketplace contract. The push oracle updates cost is shared across subscribed readers, making it cheaper for the UDF push oracle user base wanting continuous access.

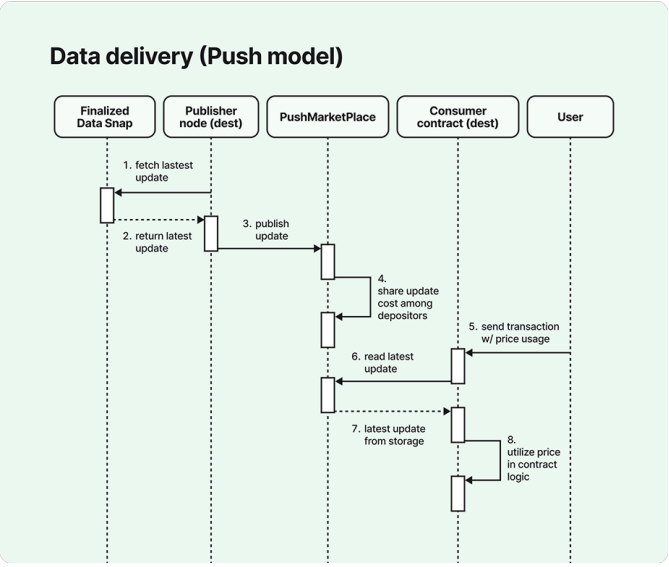


*Figure 7: Push Model Workflow – Scheduled Updates & Cost-Sharing Mechanism*

**Advantages & tradeoffs:**

• **Gas Efficiency for Consumers:** Price reads are restricted to subscribed users and data updates are shared across multiple users, reducing individual verification costs.
• **Cost Efficiency for Consumers:** Cost sharing mechanisms amongst subscribed users makes subscriptions cost efficient.
• **Slight delay in data freshness:** Updates happen periodically (price deviation triggers or time-based heartbeats).
• **Best for continuous data access:** Suited for DEXs, stablecoin issuers, and liquidity protocols needing steady price updates.

### 4.3 Oracle Contract & User Interaction

UDF's oracle contracts provide two distinct interfaces, Pull and Push, designed to deliver flexible data access for dApps. The Pull interface enables on-demand verification within user transactions, supporting both Pay-As-You-Go (PAYG) and subscription-based models. The Push interface offers subscribed users seamless access to pre-verified, on-chain data updates. These interfaces, detailed in the table below, leverage VerificationLib to ensure data integrity and adaptability across diverse use cases.

| **Pull** | |
|---|---|
| **Function** | verifyWithFee() |
| **Purpose** | Processes the protocol fee for the Pay-As-You-Go (PAYG) pull model during a transaction, verifies transmitter votes, and returns the aggregated data value. |
| **Usecase** | Ideal for dApps with sporadic data needs. |
| **Function** | verifyAsSubscriber() |
| **Purpose** | Confirms the user has an active pull subscription during a transaction, verifies votes, and returns the aggregated data value without additional protocol fees. |
| **Usecase** | Suited for protocols (e.g. lending) needing frequent, real-time price checks with predictable costs. |
| **Push** | |
| **Function** | getFeedPrice() |
| **Purpose** | Allows subscribed users to read the latest push feed data stored on-chain without verification overhead. |
| **Usecase** | Suited for DEXs or stablecoin issuers requiring continuous price updates for liquidity pools. |

### 4.4 Data Endpoints & Omnichain Infrastructure

• **Multiple Chains:** Each supported chain has UDF Oracle Push and Pull contracts deployments.
• **Push Node Authorization:** Authorization mechanisms manage the push nodes that are permitted to submit updates on each chain, ensuring controlled data propagation through access control mechanisms.
• **Cross-Chain:** If a protocol operates on multiple chains, EIB bridging can propagate the final snapshot from the aggregator through the EIB and towards multiple chains, ensuring uniform data across all deployments.

**Omnichain Messaging & Cross-Chain Data Propagation**

UDF ensures seamless omnichain data propagation by leveraging Entangle's cross-chain messaging protocol. Transmitter nodes monitor the Entangle Interoperable Blockchain (EIB) to efficiently relay oracle data across different networks. This setup enables push updates to be mirrored on multiple chains simultaneously, reducing inconsistencies and ensuring real-time availability of oracle data for multi-chain dApps.

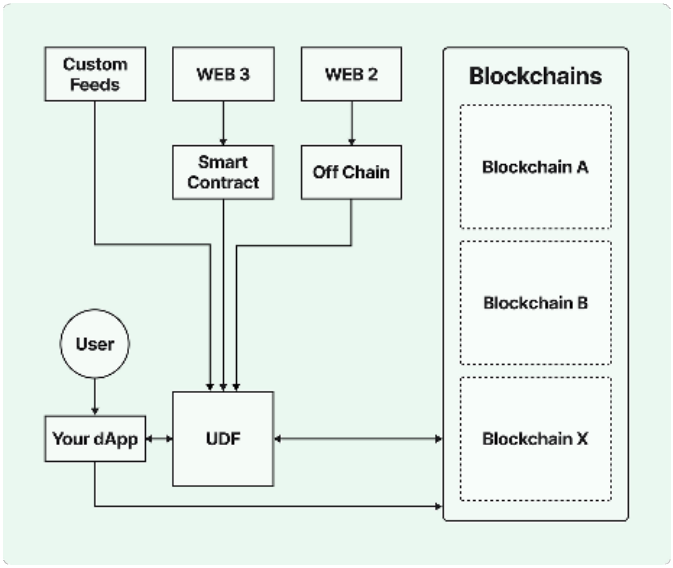


*Figure 8: Omnichain Data Flow – Cross-Chain Oracle Propagation via EIB*

# 5. Pricing Model

UDF's flexible monetization covers both subscription-based payments for push and pull feeds and a pay-as-you-go (PAYG) fees model for pull verifications.

### 5.1 PAYG vs. Subscription

UDF offers flexible pull and push pricing models, allowing users to choose between Pay-As-You-Go (PAYG) and Subscription plans based on their data needs and cost preferences.

**Pull Model**

• **PAYG (Pay-As-You-Go):** Users pay a small protocol fee per request, including the aggregator's proof in their transaction. This model is best for dApps with sporadic or low-frequency data needs, as costs scale with usage.

• **Subscription:** Users or protocols subscribe to continuous data access for a fixed periodic (e.g. monthly) fee, eliminating per-request costs. This model is ideal for applications with consistent data consumption, reducing transaction overhead while ensuring predictable expenses.

For PAYG pull oracle users, the total cost per verification includes gas fees for verification and a protocol fee:

$$C_{pull,\,payg} = G_{verify} + F_{protocol}$$

Where:

$C_{pull,\,payg}$ = Total cost for verifying oracle data
$G_{verify}$ = Gas cost for on-chain verification
$F_{protocol}$ = Protocol fee charged per data feed request

This model is more cost-effective for protocols requiring frequent or real-time oracle data access.

**Pull Subscription Model (Fixed Fee)**

For pull subscriptions, users purchase monthly subscriptions and only pay for the on-chain verification gas when performing pull updates. This eliminates the protocol fee component, allowing verifications without protocol fee:

$$C_{pull,\,sub} = G_{verify}$$

Where:

$C_{pull,\,sub}$ = Total cost for verifying oracle data
$G_{verify}$ = Gas cost for on-chain verification

This model is more cost-effective for protocols requiring frequent or real-time oracle data access.

**Push Model**

• **Fair Pricing Model:** Users prepay for updates, covering gas costs upfront while a small percentage-based fee is applied at the time of deposit. The protocol ensures pricing transparency by dynamiclly estimating costs based on gas prices and cost sharing mechanisms.
• **Exclusive Access:** On-chain price data is restricted to paying users, ensuring that only those contributing to the ecosystem benefit from updates.
• **Cost Sharing Mechanism:** As more consumers subscribe to a feed, the cost per user decreases, making the model more economical at scale.

**Cost Sharing Mechanism**

To maintain fair pricing, UDF employs a cost-sharing formula among subscribed users:

$$C_{total} = \sum_{i=0}^{N_{subs}} C_i$$

Where:

$C_i$ = Cost per subscriber
$C_{total}$ = Total oracle update gas cost
$N_{subs}$ = Number of users sharing the feed

This ensures that as more users subscribe to the same data feed, individual costs decrease, making it more affordable.

Additionally, a deviation-weighted cost distribution approach ensures that users with stricter update frequency (lower deviation thresholds) contribute proportionally more to oracle update costs:

$$w_i = 24 - h_i + \frac{\alpha}{d_i^3}$$

Where:

$w_i$ = Weight of user i
$h_i$ = Heartbeat selected by user i
$\alpha$ = Scaling coefficient
$d_i$ = Deviation threshold set by the user i

The Figure 9 below shows how user weights are assigned based on deviation and heartbeat settings, ensuring that users with stricter update frequencies bear a proportionally higher cost. It shows how stricter settings increase a user's cost share.

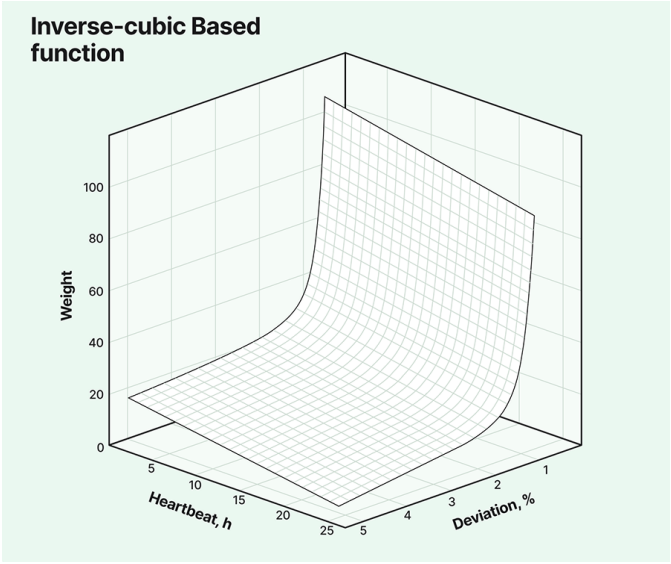


*Figure 9: Weight Allocation – Impact of Deviation and Heartbeat on User Weights*

This ensures that users requiring frequent updates pay more, while those with wider deviation tolerances contribute adequately less, maintaining fairness and economic efficiency. The cubic factor was chosen since update frequency scales non-linearly with deviation, requiring a steep weighting function to maintain fair cost distribution.

Each user's final contribution to the oracle update gas cost is determined proportionally based on their assigned weight. The total fee paid by an individual user is computed as:

$$C_i = \frac{w_i}{\sum_{i=0}^{N_{subs}} w_i} \times C_{total}$$

Where:

$C_i$ = Gas cost component paid by user i
$C_{total}$ = Total oracle update gas cost
$w_i$ = Weight of user i, calculated based on deviation and heartbeat parameters
$\sum_{i=0}^{N_{subs}} w_i$ = Sum of $N_{subs}$ users' weights sharing the same push feed

The cost-sharing mechanism in UDF dynamically adjusts based on users' selected deviation thresholds, ensuring fair distribution of oracle update costs. This mechanism prevents users with wider deviation thresholds from incurring higher costs when others opt for stricter deviations.

By supporting both pull and push subscriptions, UDF delivers a scalable, cost-efficient oracle service designed for diverse dApp needs.

### 5.2 Cost Structure Across Multiple Chains

Gas prices vary significantly (e.g., Ethereum vs. an L2) and UDF must adapt. Fees vary based on the target chain's gas costs; higher fees apply to expensive chains, while cheaper chains benefit from lower costs. Pricing is periodically adjusted to reflect changes in network fees, ensuring sustainable operations across all supported chains.

## 6. Customizability & Future Directions

**1. Custom Aggregation Logic:** Some dApps might prefer Volume-Weighted Average Price (VWAP), Time-Weighted Average Prices (TWAP) [7], median values, or multi-data-key correlation checks. UDF's aggregator can be tailored for such use cases.
**2. TEE Integration:** Enabling privacy-preserving or confidential data processing is possible through Intel SGX or other TEEs enclaves [8]. This ensures zero-knowledge style computations within aggregator nodes and enables real time data feeds with ultra low latency.
**3. Multi-Signature Threshold:** dApps can specify how many transmitters must sign (and from which set) to consider a data feed valid.

Developers and users can explore these features and visit the UDF Portal [9], where they can access real-time data feeds, configure subscriptions, and manage oracle integrations. For technical details and implementation guides, refer to the UDF Documentation [10] and Developer Guides [11].

## 7. Conclusion

UDF redefines oracle performance and delivers a scalable, secure oracle standard for the multi-chain future with hyper-fast, cost-efficient, and secure data feeds, setting a new industry standard for real-time, trustless information across multiple blockchains. By combining:

- **Pull feeds** for cost-efficiency, minimal latency, and transaction-level verification
- **Push feeds** for automated, high-frequency updates at scale
- **Omnichain compatibility** for seamless integration across EVM and non-EVM ecosystems
- **EIB-powered security** with Byzantine fault-tolerant consensus and scalable finalization
- **Decentralized transmitters** ensuring reliable, tamper-proof data validation

UDF eliminates traditional oracle trade-offs, reducing complexity and overhead in both cross-chain and single-chain solutions. By optimizing cost, security, and speed, it empowers developers to choose the most efficient data model based on their dApp's usage patterns and cost sensitivities, ensuring seamless integration and dynamic scalability.

Looking ahead, UDF will integrate TEE-powered efficiency for ultra-fast finalization while advancing aggregation models and cross-chain expansion. Positioned as the backbone of future internet data infrastructure, UDF will bridge DeFi, AI, RWA, and beyond.

## 8. References


1. A. Alamsyah, G. N. W. Kusuma, and D. P. Ramadhani, "A review on decentralized finance ecosystems," Future Internet, vol. 16, no. 3, p. 76, 2024. [Online]. Available: https://www.mdpi.com/1999-5903/16/3/76
2. LayerZero, "LayerZero Whitepaper," [Online]. Available: https://layerzero.network/publications/LayerZero_Whitepaper_V2.1.0.pdf
3. Arbitrum, "Optimistic Rollup," [Online]. Available: https://docs.arbitrum.io/how-arbitrum-works/optimistic-rollup
4. S. Nakov, "ECDSA: Elliptic Curve Signatures" Practical cryptography for developers, [Online]. Available: https://cryptobook.nakov.com/digital-signatures/ecdsa-sign-verify-messages
5. Entangle, "UIP Whitepaper," [Online]. Available: https://entangle.fi/documents/UIP%20Whitepaper.pdf
6. Ethermint, "Ethermint Documentation," [Online]. Available: https://docs.ethermint.zone/
7. Halborn, "What Are TWAP Oracles?" Halborn Blog, [Online]. Available: https://www.halborn.com/blog/post/what-are-twap-oracles
8. Intel, "Trusted Execution Environments (TEE)" [Online]. Available: https://docs.trustauthority.intel.com/main/articles/concept-tees-overview.html

9. Entangle, “UDF Portal,” [Online]. Available: https://udf.entangle.fi
10. Entangle, “Universal Data Feeds: Overview,” Entangle Documentation, [Online]. Available: https://docs.entangle.fi/universal-data-feeds/overview
11. Entangle, “Universal Data Feeds: Developer Guide,” Entangle Documentation, [Online]. Available: https://docs.entangle.fi/universal-data-feeds/developer-guide