Zenith Lightpaper, V0.2

Preamble

Below is an outline of Zenith, a global consensus layer that provides Urbit with an economic substrate that we believe will accelerate the adoption of Urbit as the de-facto, full stack decentralized operating system and marshall it to wide scale adoption.

We frame our views on the architecture of Zenith, the design of a fungible Urbit token ($Z), a roadmap that includes an incentivized testnet meant to allow for experimentation with extant Urbit applications (namely, Tlon Messenger), a launch mechanism called a lockdrop meant to distribute economic interest to current Urbit network participants, and the product possibilities that this approach unlocks.

Abstract

Zenith integrates natively with the Urbit network by designating Urbit's infrastructure nodes — Galaxies and Stars — as blockchain validators through a hybrid Proof of Authority/Proof of Stake mechanism. This architecture, combined with enshrined proposer-builder separation, ensures decentralized consensus while regularizing MEV extraction. Zenith allocates nonfungible blockspace to Stars, creating both a predictable fee market and a decentralized marketplace for blockchain services. The system's native currency $Z enables direct economic activity within the Urbit network, while its decentralized global state provides the foundation for social coordination. Through the novel Scry Oracle system, Zenith leverages Urbit's peer-to-peer computing infrastructure to create a mechanism for managing global state built on top of Urbit’s static functional namespace. This enables Urbit applications to implement features requiring shared state—from reputation systems to asset to collaborative tools—without complex external dependencies. By storing only cryptographic commitments on-chain while delegating computation and data storage to the network edge (Urbit), Zenith achieves sub-second transaction finality with the potential to process millions of state updates per minute. This architecture combines the security guarantees of blockchain consensus with the performance characteristics of traditional distributed systems, enabling real-world applications that would be impractical on conventional smart contract platforms.

Introduction

What is Urbit and why Zenith?

Urbit is a from scratch reimagining of networked personal computing, designed to give users complete ownership over their digital lives through self-sovereign identity, data, and compute. It aims to create a paradigm where users control all of their data and applications and interact

without a reliance on trusted third parties.To fully realize this vision, Urbit must solve two fundamental challenges inherent to decentralized computing that were out of scope at the time of its creation: establishing shared truth and enabling sovereign value capture by network participants through a mechanism native to the Urbit network.

Zenith introduces three capabilities that transform Urbit from a network of sovereign computers into a complete platform for decentralized social computing:

First, it implements a hybrid Proof of Authority/Proof of Stake consensus mechanism that leverages Urbit's existing infrastructure nodes, creating a naturally decentralized blockchain that can be secured by the network's existing Galaxy operators. This system incorporates enshrined Proposer-Builder Separation (ePBS) by giving Urbit Stars guaranteed access to blockspace, enabling them to provide specialized blockchain services and be remunerated in a currency native to the network while maintaining decentralization.

Second, it implements the Scry Oracle system, a novel mechanism for managing global state built on Urbit's static functional namespace. The Scry Oracle adds Byzantine Fault Tolerance to Urbit's existing “scry[1]” namespace, making any scry binding immutable, canonical, and discoverable. This enables applications to implement features requiring shared state—from reputation systems to asset ownership to collaborative tools—without complex external dependencies.

Finally, it introduces $Z, a native fee token that serves as a foundation for economic activity within the Urbit network. Beyond simple payments, $Z provides primitives that enable a rich set of economic interactions, many of which have been experimented with in extant blockchain systems, and many of which are wholly unique and only possible because of a tightly integrated system, ie reputation systems, digital asset ownership, “light” contracts, and service provision, tightly integrated into a first of its kind sovereign computing platform.

Background

Urbit began in 2002 as an independent research project derived from three key insights about the failures of networked computing: that Unix-based personal computing had become impersonal, with users owning none of their networked data; that the Internet's architects never anticipated the need for sovereign user-owned digital identity mapped to personal compute and storage; and that the client-server model fundamentally limited software extensibility and interoperability. The creators of Urbit foresaw a future where the user of networked services would be ruled over by whoever ran the server.

This motivated the development of Urbit’s two core technologies: Azimuth, a Public Key Infrastructure (PKI) of 2^32 (~4.3 billion) network addresses that serve as user identities, and Arvo, a deterministic operating system designed for personal ownership of software and data over a P2P network. These components serve as the foundation for a world where any user can

[1] A scry path is basically just a file path pointing to some internal data on an Urbit ship

run their own server, which is the necessary precondition for a sovereign social operating system.

Early Development

When Urbit was first conceived, blockchain technology did not exist to solve problems of decentralized consensus and economic coordination. Even by 2015, when Ethereum launched, the capabilities of programmable blockchains were not yet well understood. As a result, Urbit succeeded in creating a system for personal sovereignty over compute and storage but lacked mechanisms for establishing shared truth across the network—a capability that would later become available through blockchain technology.

The Current State of Urbit & Challenges

Today, the Urbit network supports thousands of nodes and has demonstrated capacity to scale to hundreds of thousands. Since 2021, hundreds of developers have created applications on Arvo, and Tlon Corporation has leveraged it to build a production-grade cross-platform group messenger app that is available in major app stores.

The Need for Consensus

Urbit faces two significant technical limitations that blockchain technology is now capable of addressing:

Lack of Global State

The lack of protocol-level global state presents a fundamental challenge for application developers on Urbit. For example, a global social graph is necessarily part of a useful social application. The status quo requires a centralized third party to store the graph, but this is both a liability to the individual “owner” of the graph itself and incurs exponentially increasing (n^2 based on graph size) opportunity cost on the user insofar as they have effectively traded the asset to a third party in exchange for their usage of the product. External dependencies introduce their own challenges around censorship resistance, performance, reliability, and developer experience—and more importantly, they fragment the network's activity across multiple systems. A decentralized computing platform requires not just the ability to track how many nodes exist and who owns them[2], but also to maintain a canonical record of how these nodes interact and connect with each other.

External solutions also typically require complex integration work and new dependencies, creating additional barriers for developers and degrading user experience. This complexity is particularly acute in blockchain integrations, where impedance mismatches between smart contracts and application code often result in disjointed user experiences and slower development cycles.

[2] This is all that Urbit uses BFT consensus for at present.

Lack of Native Economic Primitives

While Urbit's architecture enables users to own their data and run their own software, it requires native economic primitives to create sustainable incentives for network growth and development. New mechanisms are needed to enable direct value capture by users and service providers while maintaining the network's decentralization without creating external dependencies.

In practice: Building a Decentralized OS with the Current State of the Art

Looking at the challenges of building a decentralized operating system without Zenith and Urbit's integrated approach reveals why previous attempts have struggled. Doing so with extant technologies requires stitching together fundamentally misaligned technologies, creating a patchwork system with critical seams that compromise the entire architecture. To understand the depth of this challenge, we need to examine what happens at each layer of the required stack.

Traditional approaches begin by attempting to decentralize computing through various disconnected solutions. Developers might leverage distributed compute networks like Golem, implement container orchestration across volunteer nodes, or create peer-to-peer virtual machines with limited guarantees. However, these approaches immediately encounter fundamental problems. Without a unified identity model across computational resources, the system suffers from unpredictable performance and availability. Security vulnerabilities emerge at system boundaries where different technologies meet, while high latency from coordination overhead makes real-time applications impractical. Most critically, the inability to guarantee deterministic execution across nodes means that different parts of the network may compute different results from the same inputs.

Building an operating system layer on top of this fragmented compute foundation only multiplies the complexity. Developers must create custom protocols for resource management while integrating disparate storage solutions like IPFS, Filecoin, or Arweave. They need to implement peer-to-peer networking stacks such as Matrix, Secure Scuttlebutt, or libp2p, while also managing replicated state machines through consensus protocols like Raft or Paxos. This results in a fragmented user and developer experience where each component requires specialized knowledge of its unique API surface. The inconsistent security models across these components create additional vulnerabilities, while the high resource requirements for running full nodes limit participation and threaten decentralization.

The challenge of maintaining consistent state across such a decentralized system introduces yet another layer of complexity. The system suffers from growing state bloat that threatens decentralization as more data accumulates on-chain. Inefficient synchronization mechanisms and inadequate conflict resolution strategies lead to slow convergence on shared state, making the system feel sluggish and unresponsive to users.

When financial capabilities need to be added to this already complex stack, they're typically bolted on through external systems rather than integrated naturally. Smart contract platforms like Ethereum or Solana provide the computational layer for financial logic, but require cross-chain bridges for interoperability with the rest of the system. External oracles become necessary to bring off-chain data into the financial layer, while Layer 2 scaling solutions attempt to address transaction throughput limitations. This creates incongruent programming models where developers must context-switch between the financial substrate and the operating system layer. Users face expensive and unpredictable transaction costs, while liquidity remains fractured across multiple chains. The complex key management required for interacting with multiple systems creates significant usability challenges, and the poor composability between financial and operating system features limits what applications can achieve.

At the application layer, developers must accommodate all this underlying complexity, working across multiple development environments and languages while managing different deployment targets for operating system versus financial functionality. Custom state management patterns become necessary to bridge the gaps between layers, while hybrid authentication models attempt to unify the disparate identity systems. This extreme complexity raises barriers to entry so high that only the most determined developers can build meaningful applications. Users experience this as inconsistent interfaces across applications, poor discoverability of services, and limited composability between different tools. The difficulty of building network effects within such a fragmented ecosystem means that even well-designed applications struggle to achieve critical mass.

The end result is a system where developers spend the majority of their time managing integration points between disparate technologies rather than building valuable functionality for users. Users, in turn, face a fragmented experience requiring them to manage multiple identities, wallets, and interfaces just to accomplish basic tasks. The overall system fails to deliver on the promise of a truly decentralized operating system with seamless financial capabilities, instead offering a complex assemblage of parts that never quite work together as intended. This is precisely the problem that Zenith solves by providing native consensus and financial primitives directly integrated into Urbit's unified computing environment.

Tying It All Together: Zenith Protocol

Zenith is a blockchain protocol that maps its consensus roles to Urbit's existing network hierarchy. It offers **strong decentralization** through Urbit's existing Galaxy distribution, which already has over 100 independent owners (no single entity controls more than 25% of total supply). The protocol provides **built-in Sybil resistance** through Urbit's identity system, enables **predictable transaction costs** through guaranteed blockspace allocation, and **handles MEV (Maximal Extractable Value) elegantly** by explicitly enshrining it within the existing hierarchy. It also **integrates seamlessly with Urbit's infrastructure**—any Urbit application can initiate blockchain transactions without external dependencies or additional key management.

The protocol's architecture centers on three key elements built around Urbit's existing node hierarchy. First, decentralized consensus is achieved through Galaxies—the root nodes of the Urbit network—which serve as validators. Second, blockspace is managed by Stars—Urbit's network relays—which act as service providers and block builders. Each Galaxy receives a fixed amount of blockspace for its sponsored Stars, enabling them to provide reliable blockchain services while discouraging any individual high-volume activity from monopolizing network resources. Third, the Scry Oracle system adds Byzantine Fault Tolerance to Urbit's "scry" namespace, creating a secure foundation for applications that need canonical, verifiable data.

Building a blockchain designed to be used by the Urbit network unlocks these features with much less work than by building on existing chains, because of the ability to only build the necessary features and primitives required to bootstrap a Zenith enabled Urbit economy and not having to cater to constituencies outside the Urbit network. For example, the Urbit network can control its own fee market, rather than being subject to competition for blockspace against memecoins and other projects. Similarly, the Scry Oracle Contract when enshrined in the consensus layer can take advantage of its inherent scaling properties, ***reaching millions of bindings per minute without novel research (back of the envelope: one 50MB block could store 100,000 scry bindings, and the system could process one block every two seconds, meaning 3 million bindings per minute) – reaching comparable performance on existing chains would be exceedingly difficult if not impossible.***

Network Architecture

In Zenith, each tier of the network serves a specific role:

Galaxies (Urbit root nodes) serve as validators and block proposers. Stars (Urbit network relays) manage chain resources by serving as block builders and service providers. Planets (Urbit personal nodes) submit transactions to Stars.

The network structure reflects the inherent scarcity and responsibilities of each tier:

	Galaxy	**Star**	**Planet**
Address Size (bits)	8	16	32
Supply	256	65,280	4,294,901,760
Sponsor	None	Galaxy	Star
Urbit role	Governor	Service provider	End-user
Zenith role	Block Proposer	Block builder	Transaction submitter

Consensus Mechanism

Galaxies serve as validators, responsible for proposing and validating blocks. This role builds on their existing position as the ultimate trust roots of the Urbit network, creating a hybrid Proof of Authority/Proof of Stake system where block confirmation rights are limited to Galaxies (Authority) who maintain a minimum stake requirement (Stake). Each validating Galaxy must maintain a minimum stake of $Z to participate in block production. All participating Galaxies meeting this stake requirement are assigned slots in a stake-weighted round-robin schedule for block production.

The stake requirement serves primarily as a security deposit against misbehavior: failing to produce a block in an assigned slot results in slashing, which reduces both the Galaxy's stake and its likelihood of receiving future slots. Repeated failures have compounding consequences—if a Galaxy misses more than 5% of their assigned blocks, they are suspended from the validator set for some number of blocks. This strict performance requirement, combined with the loss of potential fee revenue from future blocks, creates strong incentives for reliable block production.

Enshrined Proposer-Builder Separation (ePBS)

Proposer-builder separation refers to the separation of block “proposing” from block “building”. “Enshrined” PBS enforces this separation at the protocol level.

Zenith's ePBS mechanism ensures every Galaxy has an opportunity to include transactions from its own Stars, while also elegantly addressing MEV by explicitly enshrining it within the protocol: Stars capture local MEV from their planets, Galaxies capture local MEV from their stars, and global MEV rights rotate between block producers in a fair and predictable fashion.

Stars, as "builders", build "bundles" of transactions, which they receive from Planets and send to Galaxies. Galaxies, as "proposers", create blocks out of groups of bundles. Those blocks are then subject to validation by other Galaxies.

For a given block, a Galaxy is selected to be the leader through a pseudorandom process that selects it from the set of Galaxies who have announced their participation and have enough $Z staked. Every block has a maximum size limit, in bytes. The first and last transaction bundles of each block must come from the leader Galaxy's sponsored Stars in order for the other Galaxies to consider it valid. This gives each Galaxy guaranteed access to a certain amount of blockspace, which it can divvy up among its Stars as it sees fit.

The leader Galaxy can decide which other transaction bundles it includes in between the first and last bundles. Galaxies may choose to share transaction bundles from stars among each other and include bundles from stars that are not bonded to them.

Stars dictate the requirements for which transactions are included in blocks and they may charge planets in any denomination (ie $Z, USDC, etc) however they want. Galaxies price

bundles in $Z, the consensus algorithm requires that galaxies pay their fees in $Z but galaxies are able to charge Stars however they want.

Network Dynamics Under Contention

When not under contention, transactions proposed by any Planet will go from their sponsoring Star to that Star's Galaxy, as part of a bundle from that Star, then the Galaxy will forward a collection of bundles to the leader forming that particular block.

Under contention, the leader forming the block is going to bias its own Stars, and within that it will generally bias Stars who have paid the most, according to that Galaxy's own decision making. The leader Galaxy will also opportunistically accept bundles signed by other Galaxies, these bundles necessarily include transaction bundles from Stars that are not its own sponsees. There are no formal constraints on which bundles the leader galaxy is required to accept from other galaxies. The general idea is that a Galaxy will take whichever bundles make it the most money. Stars are guaranteed blockspace, since Galaxies have to include certain bundles from the Stars they sponsor, but Galaxies have no requirement to include any bundles from any other Galaxy or a Star they don't sponsor.

Network Participant Roles

Stars

Stars function as block builders, managing chain resources by accepting transactions from Planets and bundling them for Galaxies to propose for consensus. Stars must stake $Z to their sponsoring Galaxy to cement this relationship and gain access to guaranteed blockspace. To prevent switching between Galaxies casually or as part of attempts to game the system, this relationship is bonded on both sides. Both the Star and the Galaxy will be slashed if either side breaks the relationship, incentivizing stability.

Each Galaxy receives a fixed allocation of blockspace that it can only distribute among its own Stars, enabling predictable transaction throughput. This predictability is crucial: rather than dealing with variable costs from competing in general blockspace auctions, Stars can reliably plan how to use their allocation. A Star might choose to:

- Offer fixed-price transaction services to their Planets, knowing exactly how many transactions they can process
- Subsidize certain types of transactions for their sponsored Planets as a way to attract users
- Specialize in specific transaction types like PKI operations or Scry Oracle services
- Operate Layer 2 environments with predictable base layer costs

This guaranteed blockspace model particularly benefits service providers who need reliable transaction processing. For example, a Star operating a payroll service can make a deal with its

Galaxy to always have first rights to a part of the Galaxy's guaranteed blockspace, letting it calculate exact costs based on its known blockspace allocation, rather than dealing with unpredictable fee spikes from network congestion. This predictability allows Stars to develop sustainable business models and offer reliable services to their users.

The role of block builder builds naturally on Stars' existing position as service providers in the Urbit network, where they already offer services like cloud storage, backup services, and network relays to their sponsored Planets.

Planets

Planets serve as end-user nodes and wallets, capable of signing and sending transactions directly over the Urbit network. This builds on their existing role as personal servers, requiring no additional configuration beyond acquiring funds to interact with Zenith.

Planets are also a Sybil-resistant identity system. This means contracts that use Planet IDs can lean on the fact that creating new identities has some associated cost, which makes reputation easier to develop.

Planets have four-syllable names like ~morzod-ballet or ~rovnys-ricfer, making them easy to remember. Planet names map to Zenith addresses, making human exeuction of peer to peer payments less frictionful on the network.

Special Considerations

Galaxy-Owned Stars

In practice, each Galaxy will want to be running at least one sponsored Star for submitting its own transactions. This keeps the layering clean, so that only Stars ever sign transaction bundles. This is a part of the design of ePBS, where the proposing and building entities must be cryptographically distinct, but not necessarily owned by distinct entities. There would be no way to enforce any policy intending to prevent a Galaxy and Star from colluding, so a robust protocol needs to be designed to function properly in the case of collusion.

As an example of a malicious version of collusion, if a Galaxy owner also owns a Star under that Galaxy, then that person could include only their own transaction bundles and censor the other sponsored Stars under that galaxy. If they do that, though, their other sponsored Stars will escape to other Galaxies, and this Galaxy owner will lose all the revenue from them. It is likely that those Stars had either expertise or order flow that the Galaxy will lose access to – because the Stars are specialized labor, it doesn't make sense for the Galaxy to compete with them.

Azimuth Transactions

In a world in which the Urbit PKI would move to Zenith, Zenith could prioritize Azimuth transactions, to ensure those never get priced out by other transactions. It could do this by

processing Azimuth transactions first – a Galaxy would take the block of transactions, separate it into Azimuth transactions and other transactions, process the Azimuth transactions first, then process other transactions. If an Azimuth transaction invalidates another transaction within the same block, the non-Azimuth transaction does not perform whatever state update it would have, but the submitter of that transaction must still pay the transaction fee.

The chain could fix the price of Azimuth transactions. Separating out the Azimuth transactions and running them first prevents various games by creating an immutable PKI within any given block – this prevents things like someone trying to raise the price of something to prevent an address ownership transfer, or extracting more money from that.

The Scry Oracle System

Understanding Urbit's Scry Namespace

The scry namespace is Urbit's native system for publishing and accessing data at a particular scry path, functioning like a global decentralized filesystem. When you want to retrieve data from another Urbit node, you use a scry path — similar to a URL, but starting with an Urbit ID instead of a domain name. Any node can request data from another node by sending a request over the Urbit network at a path that starts with that node's ID.

When an Urbit node responds to a scry request, it signs the response with its private key[3]. This signature serves two purposes: it proves the response came from the correct node, and it represents a promise by that node to always return the same data for that path. The node that published the scry binding is generally responsible[4] for serving the data at that path, although anyone is free to cache and republish the data.

Adding Byzantine Fault Tolerance

While nodes promise to return consistent data, the Urbit network currently cannot enforce this promise. A node could send different data to different requesters for the same path — a "double bind" problem analogous to the "double spend" problem that blockchains solve. Mathematically, this constitutes a Byzantine fault.

A large amount of economic value can be unlocked by providing resiliency against Byzantine faults. In many applications in which multiple parties need to coordinate on a single canonical version of the truth – often the case when the authenticity and immutability of data has financial or legal implications – the cost of BFT consensus is justified.

For example:

[3] Technically, this is the “networking key”, which is one of several keys in Urbit’s BIP-32 compliant HD wallet system – distinct from the key that owns the address, which should be kept in cold storage
[4] But not required. We’ll discuss this more further below.

Coordination Around Truth

- Oracle services must ensure all participants see identical price data
- Reputation systems require consistent display of ratings and reviews
- Supply chain participants need agreement on tracking information

Financial and Legal Implications

- Financial institutions must maintain immutable audit trails
- Healthcare providers need verifiable records that meet regulatory requirements
- Different jurisdictions may require proof of data integrity and chronological ordering
- Legal documents require verifiable timestamps and proof of authenticity

Removing Intermediaries

- Content creators need to prove ownership without central registries
- Credential issuers must maintain consistent records without relying on third parties
- Identity systems require decentralized verification of attestations
- Reputation systems must function without centralized rating authorities

The Scry Oracle adds Byzantine Fault Tolerance to the Urbit scry namespace, solving the double-bind problem described earlier. It maintains an on-chain registry of cryptographic hashes that correspond to data published at specific paths on Urbit nodes. When a node publishes data, the Oracle:

1. Verifies the publisher owns the path
2. Checks that the path hasn't already been bound
3. Records the hash of the data

The Scry Oracle is a registry that guarantees the canonical value for any path, i.e. the piece of data a node would send in response to a network request for that path. This binding between path and value must be signed, so that anyone could verify objectively that the node had indeed published the binding, not someone else. When hearing about a new binding, the registry also needs to abort the transaction if the path had already been bound. The registry only stores the hashes of the values at scry paths, not the values themselves, which can be arbitrarily large — its purpose is solely canonicalization, not data storage. Data storage is already handled by the Urbit network, off-chain.

The Scry Oracle enables the above applications without requiring complex smart contracts. It provides a simple primitive — canonical data binding — that can be composed into sophisticated systems while maintaining Urbit's decentralized nature.

On the surface, the SOC is a simple decentralized check-sum registry. Extending it with economic incentives to add guarantees around data availability at registered bindings would

enable use cases like decentralized app stores and file sharing networks. That is out of the scope of this paper, but we expect these to be built either internally or relatively quickly by independent teams.

For more details, see the Developer Experience section below.

Adding Discoverability to Scry Bindings

Without the Scry Oracle, scry bindings are only discoverable insofar as the publisher makes the path known to other nodes. With the Scry Oracle, every binding is globally available in a single location, the Scry Oracle contract, making it possible for nodes to discover what data has been published.

This allows applications to run global queries and aggregations on the full set of canonicalized scry bindings. Use cases include app stores, payment gated content, order books, leaderboards, reputation systems, and content curation. In essence, the Urbit network becomes a decentralized filesystem with on-chain listings and off-chain data storage.

Developer Experience

Zenith represents a fundamental shift in how developers can build blockchain-enabled applications. Rather than treating on-chain and off-chain components as separate systems that must be carefully integrated, Zenith allows developers to treat consensus state as a natural extension of their application's data model.

As Urbit's native consensus system, Zenith will provide a highly ergonomic API to Urbit developers. This integration manifests in several ways:

1. **Native Data Access:** The Urbit OS makes data from the scry namespace available axiomatically to applications – it can be accessed synchronously and without violating the referential transparency of Nock[5] programs, just like data in the program's local variables.
2. **Simplified Consensus:** Applications can request data canonicalization through simple one-line API calls, allowing Urbit application developers to focus on building features rather than managing blockchain interactions.
3. **Unified Identity:** Zenith contracts can use the Urbit PKI natively, eliminating the need to manage separate on-chain and off-chain identity systems.

This development environment enables rapid iteration and experimentation. Developers can build tightly integrated applications that combine on-chain and off-chain functionality without the traditional separation between smart contracts and "dApps."

[5] Nock is Urbit's machine code – in this context, equivalent to EVM bytecode.

Example: Providing a Rating by Canonicalizing a Scry Binding

Consider the use-case of rating posts in a blogging application. With Zenith and Urbit, no centralized server is required to store either the blog posts or ratings, since each user publishes their own posts, and each consumer publishes their own ratings. An application-specific smart contract is also not needed, because the Scry Oracle contract is expressive enough to encode all the required on-chain data.

This example demonstrates how user `~rovnys-ricfer` would provide a rating on another user's blog post — in this case `~ravmel-ropdyl`. The final line is a one-line API call to canonicalize the binding, which makes `~rovnys-ricfer`'s rating of the post globally discoverable, queryable, and uncensorable.

```javascript
var blogPostScryPath = "/~ravmel-ropdyl/blog/posts/3";
var ratingPath = "/~rovnys-ricfer/ratings" + blogPostScryPath;
var ratingValue = 4; // 1 to 5

canonicalizeScryBinding({
  path: ratingPath,
  value: ratingValue,
  onSuccess: function(signature) {},
  onFailure: function(error) {}
});
```

Also note that this blog post could be paywalled or otherwise kept private, without needing to modify this scheme. The hash posted to the chain would be the hash of an encrypted version of the blog post, and only users who have been granted access (out of band) by the publishing ship would be able to decrypt the post and view it. To reiterate: the Scry Oracle only stores the path and the hash of the canonical value at that path — the data itself is provided by the node (or nodes) that publish the data, which are free to provide that data to requesters however they wish.

Example: Aggregating Ratings by Querying the Scry Oracle

Here is some pseudocode to calculate the average rating on that blog post, by aggregating all scry bindings in the scry oracle contract whose paths match the pattern indicating they represent ratings on that post:

```javascript
// Function to calculate average rating for a blog post
async function getAverageRating(blogPostPath) {
```

```
  try {
    // Query the scry oracle for all ratings on this post
    const ratingBindings = await queryScryOracle({
      pattern: `/**/ratings${blogPostPath}`
    });

    if (ratingBindings.length === 0) {
      return 0;
    }

    // Calculate average from all rating values
    const sum = ratingBindings.reduce((acc, binding) => acc + binding.value,
0);
    return Number((sum / ratingBindings.length).toFixed(2));
  } catch (error) {
    throw new Error(`Failed to calculate average rating: ${error.message}`);
  }
}

// Example usage:
const blogPostPath = '/~ravmel-ropdyl/blog/posts/3';

getAverageRating(blogPostPath)
  .then(average => console.log(`Average rating: ${average}`))
  .catch(error => console.error(error));
```

Note that in most blockchain systems, effectively aggregating on-chain data from inside an application typically requires contract-specific indexing infrastructure to be built and maintained. Making aggregations efficient will still require indexing, but this system has a few advantages:

- Scry paths have meaningful names. Typical blockchains are queryable by hexadecimal addresses and transaction hashes, which isn't useful unless you know what you're looking for. Scry paths, by contrast, contain useful metadata about what they represent.
- Creating indices of aggregate data at paths conforming to specific patterns is substantially simpler and more regular than doing so for Turing complete contracts.
- Any Urbit node can create such an index itself as part of the process of running an application.

Implementation Strategy

Initial Scope

Zenith's initial implementation prioritizes simplicity and reliability by using proven technologies. The chain will launch with three core features:

- Consensus and block production
- A native token ($Z)
- The Scry Oracle system

Zenith will use CometBFT for consensus, running in a "sidecar" process alongside validators' Urbit nodes. CometBFT is a battle-tested consensus layer that secures billions of dollars in value. It continues to be used for blockchain implementation by leading projects such as Celestia ($2.5bn FDV), Injective ($1.3bn FDV), Cronos ($10bn FDV), Binance Chain ($96bn FDV), dYdX Chain ($530mm FDV) and Berachain ($1.3bn FDV).

A modular blockchain architecture allows Zenith to replace components with Urbit-native solutions as they mature — for example, early experiments like %chain have demonstrated that Urbit-native consensus is possible; the Directed Messaging project will make Urbit's network, %ames, capable of operating at the speeds required to replace CometBFT's networking module; the Ares project will make Urbit nodes capable of storing the blockchain directly; a Nock VM in place of EVM for contract execution would require substantial development but is a tractable problem.

Zenith will launch without smart contracts to allow the development team to focus on core capabilities while maintaining simplicity and security. The Scry Oracle system provides significant programmability without the complexity of general-purpose computation. This feature set has been chosen to minimize the time required to deliver important mainnet applications.

It's worth noting that the Scry Oracle enables smart contract execution through a novel approach: rather than implementing contracts directly in the base layer, it allows Stars to operate specialized execution environments that publish state roots through the Scry Oracle. See the EVM Compatibility section below for more details.

Incentivized Testnet → Mainnet Launch

Zenith's initial scope is intentionally constrained to achieve a balance between simplicity and expressivity. Our priorities are to:

1. Bring a useful system online in a reasonably short time frame,
2. Without sacrificing power and expressivity,
3. While ensuring that our consensus and blockspace model is secure and functional.

We intend to launch Zenith's testnet by late Q3 2025, which will include all features in the Initial Scope mentioned above. From a technical standpoint this timeline is viable because:

- CometBFT gives us off-the-shelf consensus with demonstrated security
- The Scry Oracle provides a dramatically lower surface area for programmability, eliminating substantial implementation complexity

We will give the galaxies the ability to vote on other contracts that may get added to the chain (for instance, contracts related to financialization). At the outset these contracts will be EVM compatible. Over time, the galaxies can choose to move to a permissionless contract model.

The Zenith testnet will be incentivized. Those Galaxies and Stars that participate will receive token allocations proportional to their participation when the mainnet launches. This sets participants up to receive token allocations that can be used to provide the requisite initial stake (see Galaxies and Stars for more detail) for operation on mainnet. For the network, ensuring maximum participation from infrastructure nodes as early as possible helps battle-test the consensus and blockspace model, and ensures that mainnet is sufficiently decentralized.

The set of features present in the incentivized testnet is the same set that we'll launch into mainnet with. After establishing sufficient confidence in the testnet, Zenith will launch mainnet — we anticipate that this will occur in Q1 2026.

Future Work

Later phases of development will introduce additional capabilities:

1. **Permissioned Nock Contracts**: A limited smart contract environment using Urbit's native computation model. Any Nock state machine pre-approved by the Galaxies could be added as a Zenith contract, without requiring the addition of a gas model. (6 months of development time)
2. **Unpermissioned Nock Contracts**: A gas model would be developed for Nock machine code, or for a custom bytecode derived from Nock, to allow anyone to deploy their own Nock contracts to Zenith. (12 months of development time)
3. **PKI Migration**: Moving Urbit's PKI from Ethereum to Zenith. If there is consensus across the network, this could be done in stages, starting with so-called "Layer-2" Planets, then Layer-1 Planets, then Stars, then Galaxies.

The path from CometBFT to a fully Urbit-native chain will be gradual. As Urbit's operating system matures, any component can be replaced by an Urbit-native equivalent:

- **Networking**: Urbit's `%ames` network for validator communication
- **Storage**: Urbit's filesystem for blockchain state
- **Consensus**: Urbit-native BFT consensus
- **Execution**: Nock-based smart contracts

This incremental approach allows us to launch with proven technology while maintaining a clear path to deeper integration with Urbit's infrastructure.

Applications & Use Cases

Cross-Domain Applications

The shared data namespace created by the Scry Oracle enables applications that span traditional boundaries. A professional network could implement its own reputation metrics while maintaining compatibility with broader discovery systems. A content platform could develop specialized curation algorithms while allowing its content to be discoverable through other interfaces.

More sophisticated applications become possible when combining multiple execution environments. For example:

- A lending application for providing uncollateralized loans to creators could merge verifiable metrics of social capital (like sustained subscription counts and engagement patterns) with DeFi protocols to create more sophisticated credit scoring, enabling loans backed by proven audience relationships rather than traditional collateral
- A DAO could use one Star's governance-focused EVM for voting while using another's DeFi-focused environment for treasury management
- A game could combine one Star's matchmaking and ranking system with another's asset trading infrastructure

Economic Relationships

The combination of verifiable computation and cross-system reputation enables new types of economic relationships. Stars can compete to provide specialized services, with their reputation helping users make informed choices. Developers can create applications that compose services from multiple providers, relying on verified track records rather than centralized trust in deciding which developers or providers they choose.

This creates a more efficient marketplace for blockchain services, where:

- Users can choose providers based on verified performance in their specific use case
- Stars can differentiate themselves through specialized expertise
- Developers can build on reliable infrastructure without lock-in
- Value flows directly to those providing useful services

Rather than trying to build everything into the base layer, Zenith provides the minimal primitives needed to enable this ecosystem of specialized services. The Scry Oracle's role in making

computation verifiable, combined with the reputation system's ability to track reliability across different contexts, creates a foundation for continuous innovation at the services layer.

Zenith's architecture enables many types of decentralized applications, one most of the more immediate addressable areas of impact comes from unbundling the traditional social computing stack. Today's platforms bundle three distinct functions: data storage, algorithmic processing, and user interface. This bundling creates monopolies where single companies control not just users' data, but how that data is filtered, processed, and presented. Zenith enables the separation of these components into their natural layers.

Unbundling the Social Stack

Zenith unbundles three core areas of functionality that have historically been controlled by single, centralized platforms:

First, the data layer becomes public and canonical through the Scry Oracle system. Rather than being locked in proprietary databases, social data exists in a public, verifiable form that any service can build upon. Users maintain sovereignty over their data while benefiting from network-wide discoverability and interoperability.

Second, the algorithm layer becomes competitive and community-controlled. Rather than one company determining what information reaches users, multiple providers can compete to offer the best processing and filtering of public data. Communities can develop or choose algorithms that align with their values and needs, from content recommendation systems to reputation scoring mechanisms.

Third, the interface layer becomes truly customizable. Client applications can provide specialized user experiences without needing to control the underlying data or algorithms. This enables innovation in user interfaces and interactions while maintaining compatibility with the broader network.

Tlon Messenger: Unbundling in Practice

Tlon's Urbit messenger app, already available in major app stores, demonstrates how this unbundled architecture creates practical value. While the app currently provides a complete, vertically integrated product built on Urbit's infrastructure, Zenith will enhance its capabilities by explicitly separating these three layers:

- At the **data layer**, group content and user interactions can be canonicalized and made discoverable through the Scry Oracle while preserving privacy.
- At the **algorithm layer**, communities can implement their own discovery mechanisms and reputation systems.
- At the **interface layer**, the plugin marketplace planned for the Tlon app enables customizable experiences built on this shared foundation.

The Data Layer: Public, Discoverable Activity

Today, each social application must build its own user base from scratch, creating isolated communities that can't easily interact. With Zenith, Tlon groups will be able to make their public activity discoverable across the network, enabling new members to find relevant communities and content. Private groups can still maintain control over access while benefiting from network-wide discovery when desired.

For example, a group might choose to make their event announcements or resource directories publicly discoverable, while keeping discussions private. This allows potential members to find communities that match their interests without compromising group privacy.

With a canonical, decentralized reputation substrate, Group admins can gate participation and roles, incentivize behavior, and engender new sorts of digital economic interactions (i.e. unsecured or reputation based lending) based publicly legible reputation protocols enabled by the Scry Oracle.

The Algorithm Layer: Community-Controlled Discovery

The current centralized paradigm of content discovery has created a one-size-fits-all approach optimized for engagement rather than value. By keeping user data siloed, only the company that controls the data can build algorithms to surface content. This creates a fundamental conflict of interest: platforms optimize for engagement, not for user value. Zenith makes relevant user data public, allowing for competition in the algorithm layer. Communities can develop or choose algorithms that align with their values and needs, from content recommendation systems to reputation scoring mechanisms.

Competition among algorithms can lead to a more diverse and nuanced ecosystem of discovery mechanisms. Rather than a single, opaque algorithm determining what users see, Zenith enables a diversity of approaches that can be tailored to the needs of different communities. This unbundling of the algorithm layer allows for a more sophisticated understanding of quality and relevance, leading to a more valuable user experience.

This data is canonicalized through the Scry Oracle, creating a permanent, verifiable record of quality assessments that can't be manipulated or retroactively changed. The community can then develop algorithms that surface content based on the author's verified history of valuable contributions. Because these reputation signals are canonical, they're more meaningful than simple upvotes—you can verify exactly how a member earned their reputation over time.

The Interface Layer: From Messenger to Social OS

Tlon's roadmap includes evolving their messenger into a social operating system through plugins and extensions. While traditional app stores rely on centralized reputation and distribution systems, Zenith will enable a more robust marketplace built on verifiable trust.

When developers publish plugins, they will canonicalize both the code and its metadata through the Scry Oracle. This creates an immutable record of what was published and by whom, enabling users to verify the authenticity and provenance of any plugin. More importantly, as users install and interact with plugins, their usage metrics can be canonicalized as well, creating a reliable, manipulation-resistant reputation system for developers and their code.

For example, a team collaboration plugin could build trust through verified usage patterns: how many groups actively use it, how long they've used it, what features they rely on most. Because these usage metrics are canonicalized through the Scry Oracle, they can't be gamed or falsified. A developer's reputation becomes a verifiable history of providing value to the community rather than just marketing claims or easily-manipulated store ratings.

This verifiable reputation system enables more sophisticated marketplace dynamics. Users could filter plugins based on developers' track records across multiple projects. Communities could maintain curated plugin collections with transparent selection criteria. Payment systems could evolve to reflect actual value delivered, with pricing tied to verified usage patterns rather than arbitrary subscription tiers.

The combination of verified code distribution and reliable reputation tracking will enable developers to focus on creating valuable tools without having to build complex platform infrastructure. For instance:

- A research collaboration plugin could prove its widespread adoption among academic communities
- A governance plugin could demonstrate its reliable use in high-stakes community decisions
- A content moderation plugin could show its effectiveness through verified outcomes

This creates a more efficient marketplace where value flows directly to creators of useful tools, with trust built on verifiable evidence rather than centralized authority or subjective app-store reviews.

Reputation Across Systems

The unbundled architecture of Zenith enables a powerful new capability: reputation that develops across multiple roles and types of participation. Unlike traditional systems where reputation is siloed within specific platforms or use cases, Zenith will enable participants to build verifiable reputations across different types of network activity and applications.

Consider a Urbit Star operator who runs network infrastructure: when they serve data that's been canonicalized through the Scry Oracle, their reliability in doing so becomes part of their permanent record. If they consistently serve data quickly and reliably, this builds their reputation as a dependable infrastructure provider. This same operator might also build blocks on the network, where their history of building well-formed blocks further strengthens their reputation.

Their proven track record across these different roles creates a more complete picture of their contributions to the network.

Similarly, a developer building plugins for Tlon's marketplace could leverage their reputation across multiple axes. Their plugins' usage statistics and user feedback would be canonicalized through the Scry Oracle, creating a verifiable history of the popularity of their software. If they later decide to operate infrastructure services, prospective users could verify their history of reliably maintaining code and responding to user needs. This cross-context reputation makes it easier for reliable actors to expand into new roles while helping users make informed decisions about whom to trust.

Even at the community level, reputation becomes more meaningful when it spans different types of activity. A group moderator who consistently makes fair decisions (verified through canonicalized governance actions) might be trusted to help curate content for the broader network. Their reputation for good judgment in one context supports their credibility in others, all backed by verifiable evidence rather than mere claims.

This multi-dimensional reputation system isn't possible without both the technical infrastructure for tracking verifiable actions and the social context for making those actions meaningful. The Scry Oracle provides the technical foundation by making these interactions permanent and discoverable, while Urbit's identity system ensures that reputation accrues to persistent identities that can't be easily abandoned or recreated.

Future Possibilities

The combination of unbundled infrastructure and cross-system reputation will enable entirely new categories of applications. While Tlon Messenger can capture immediate value from Zenith, the architecture supports much more sophisticated systems that weren't possible in traditional bundled platforms.

Light Contracts

Light Contracts represent an area of future development, currently in the research phase and planned for implementation after mainnet launch. Light contracts push computation to the network edge (individual Urbit nodes) rather than through consensus. Unlike smart contracts that execute on every validator node and store their entire state on-chain, Light Contracts operate as deterministic functions whose inputs are canonicalized through the Scry Oracle while execution happens locally on Urbit nodes. Zenith stores only cryptographic commitments to inputs, not the computed state—similar to Urbit’s naive rollup.

Any Urbit node can execute this logic locally by using inputs canonicalized on-chain. For example, a Light Contract would canonicalize order submissions through the Scry Oracle, but order matching and state calculation happen on individual nodes. Because all nodes execute

the same deterministic function with the same canonicalized inputs, they arrive at the same state without requiring on-chain computation. This separation of consensus (what inputs exist) from computation (what those inputs produce) enables significant performance improvements.

Light Contracts eliminate certain types of dependencies between contract executions. Since each Light Contract operates on inputs that have been canonicalized through the Scry Oracle rather than reading from shared mutable state, many contracts can be processed independently. A node can execute multiple Light Contracts simultaneously when they don't depend on each other's outputs. However, when contracts do have sequential dependencies—where one contract's output becomes another's input—they must still be processed in order. This architecture enables parallel processing for independent operations while maintaining correct ordering for dependent ones. Combined with local execution, this approach scales linearly with the number of nodes that join the network because execution is decoupled from transaction verification. Because only cryptographic commitments are stored on-chain, the blockchain remains lightweight even as contracts and computational complexity scale.

Traditional smart contracts treat the blockchain as both a source of truth and an execution environment, forcing every node to compute every operation redundantly. Light Contracts recognize that consensus is only needed for inputs, not computation. Where smart contracts must optimize for gas costs and on-chain storage, Light Contracts can perform complex calculations, store large derived state, and interact with external data without gas constraints. This enables applications that aren't economically feasible on conventional platforms— real-time pricing engines, sophisticated matching algorithms —while maintaining verifiability and determinism.

Specialized Execution Environments: Stars as L3 Sequencers

The Scry Oracle's ability to make any computation verifiable enables Stars to operate specialized execution environments without requiring changes to Zenith's base layer. For example, a Star could run an Ethereum Virtual Machine (EVM) sequencer, publishing state roots through the Scry Oracle. This creates a permissionless environment where different Stars can offer varied execution environments while inheriting Zenith's security guarantees.

This capability is particularly significant at launch: while Zenith intentionally launches without native smart contract support to maintain simplicity and security, EVM compatibility through Stars enables key functionality like new token creation from day one. Projects can deploy standard ERC-20 or ERC-721 contracts through Star-operated EVMs immediately, rather than waiting for future protocol upgrades.

The unbundled nature of this approach becomes clear when we examine its layers:

- At the data layer, state roots and transaction data are canonicalized through the Scry Oracle
- At the execution layer, different Stars can implement specialized environments optimized for specific use cases

- At the interface layer, standard endpoints enable existing tools and applications to work seamlessly

This architecture allows Stars to develop expertise in particular domains. One Star might focus on running high-performance DeFi protocols, another on gaming applications, and another on identity services. Their reputation for reliable execution in these domains, verified through the Scry Oracle, helps users choose providers that match their needs.

Cross-Domain Applications

The shared data namespace created by the Scry Oracle enables applications that span traditional boundaries. A professional network could implement its own reputation metrics while maintaining compatibility with broader discovery systems. A content platform could develop specialized curation algorithms while allowing its content to be discoverable through other interfaces.

More sophisticated applications become possible when combining multiple execution environments. For example:

- A lending application for providing uncollateralized loans to creators could merge verifiable metrics of social capital (like sustained subscription counts and engagement patterns) with DeFi protocols to create more sophisticated credit scoring, enabling loans backed by proven audience relationships rather than traditional collateral
- A DAO could use one Star's governance-focused EVM for voting while using another's DeFi-focused environment for treasury management
- A game could combine one Star's matchmaking and ranking system with another's asset trading infrastructure

Economic Relationships

The combination of verifiable computation and cross-system reputation enables new types of economic relationships. Stars can compete to provide specialized services, with their reputation helping users make informed choices. Developers can create applications that compose services from multiple providers, relying on verified track records rather than centralized trust in deciding which developers or providers they choose.

This creates a more efficient marketplace for blockchain services, where:

- Users can choose providers based on verified performance in their specific use case
- Stars can differentiate themselves through specialized expertise
- Developers can build on reliable infrastructure without lock-in
- Value flows directly to those providing useful services

Rather than trying to build everything into the base layer, Zenith provides the minimal primitives needed to enable this ecosystem of specialized services. The Scry Oracle's role in making

computation verifiable, combined with the reputation system's ability to track reliability across different contexts, creates a foundation for continuous innovation at the services layer.

$Z Token Economics

The introduction of $Z incentivizes the growth of the network, providing a means of value accrual to Urbit infrastructure and a new economic primitive to build upon. An exploration of the opportunities provided by this new economic primitive can be found further below; this section is focused on:

- The role of $Z in the Urbit network
- Distribution of $Z at TGE (Token Genesis Event)
- Address space $Z claims via Lockdrop
- $Z emissions and mechanics
- The incentivized testnet

Roles of $Z

$Z will have three roles defined by the Zenith protocol:

1. **$Z will be the native fee token of the Zenith chain.** Growth of the Urbit network is conditional upon participation from Galaxies (who issue Stars) and Stars, who issue Planets. Network growth will generate transaction revenue through PKI interactions, which will in turn be captured by Galaxies and Stars: incentives aligned.
2. **$Z will incentivize the provisioning of Urbit infrastructure nodes**[6], without which Zenith would not function properly or provide sufficient decentralization guarantees. After TGE, the only source of $Z supply will be from various forms of network participation available only to owners of infrastructure nodes.
3. **Staked $Z will be required to validate blocks and propose transactions.** Galaxies will be required to maintain a stake of at least 131,072 $Z to participate in validating blocks. All validators are required to stake the same amount of $Z. As validators misbehave their stake is slashed, which reduces their likelihood of being selected to produce blocks. If the validator's stake falls below some threshold, they will be unable to produce more blocks and evicted from the validator set. Double signing will result in slashing of staked $Z, and inactivity will result in temporary suspension from selection as a validator. Stars will be required to stake $Z on their Galaxy to form an economic relationship, providing them with access to the Galaxy's blockspace.

[6] It's essentially DePIN: https://www.theblock.co/learn/299214/what-is-depin

Tokenomic Overview

- Total Supply: capped at 4,294,967,296, matching the total number of Azimuth points
- Distribution: 60% community (between lockdrop and block rewards), 12.5% to the Zenith Foundation, 12.5% team (4 year vest with a 1 year cliff), 10% to seed investors, 5% allocated to market makers
- Block Reward Emissions: 32 years with 4 year halvings

Distribution at TGE

Address Space Lockdrop

30% of total supply will be claimable by the owners of Galaxies and Stars via the Lockdrop. Mechanically, this means that an owner of a Galaxy or Star will assign the ownership of their Galaxy or Star to the Lockdrop contract — the "Custodian" contract — for a period of time of their choosing between one and five years. Importantly, the Custodian contract will be implemented such that proxy addresses can still be interacted with. This means that a Galaxy or Star owner participating in the Lockdrop would still retain the ability to spawn Planets/Stars, vote (in the case of Galaxies), validate and propose blocks on and manage all aspects of running the actual Zenith/Urbit node. Notably, while the keys associated with the Custodian contract cannot be rotated, assignment via a multisig can accomplish this.

Procedurally, the lockdrop will be announced publicly six months prior to the launch of Zenith. There will be a six month participation window during which Star and Galaxy owners can deposit their address space into a contract over an expressed window between one and five years. Depositors will be able to participate through a web UI with transparent metrics around lock rates, tokens claimed, and the ongoing time preference of participants. Those that deposit for the maximum lock period (five years) receive tokens pro-rata from a bonus pool that is sourced from the total amount of tokens that would have been allocated to all lockdrop participants had they all locked for five years.

The purpose of the lockdrop is to incentivize address space holders with the highest economic Zenith/Urbit time preference to lock for as long as possible. Locking in the contract will be irreversible[7] as this allows the protocol to grant the totality of lockdrop tokens in a locked format immediately after the participation window closes so that participants can take receipt of them before value can be ascribed to them in the open market. All claims will be subject to a one year cliff from the date the claim is made (not TGE). Galaxy points will be allocated 1% of the TGE lockdrop allocation, with the remaining 99% allocated to stars.

Each Star is entitled to receive the following amount of tokens:

[7] Without a Galaxy vote, that is.

$$Full\ Token\ Allocation\ =\ \frac{1{,}275{,}605{,}287}{Star\ Lockdrop\ Participation\ Rate \times Total\ Urbit\ Stars}$$

Each galaxy is entitled to receive the following amount of tokens:

12,884,902/ ([Galaxy Lockdrop Participation Rate][Total Urbit Galaxies])*

Those stars not participating in the five year lock will receive tokens by the following formula with a one year cliff and a linear vest thereafter:

*[Full Token Allocation] * (1- (5 - [Lock Period in Years])*20%)*

A bonus pool will accrue based on the following formula taken as the sum of the Full Token Allocation multiplied by a penalty rate (20% for each year of each Star not participating in the five year lock period). This bonus pool will be distributed to stars locking for five years, pro rata:

*Bonus Pool = [Star Lockdrop Participation Rate] * ([% of Stars Locked for 4 years] * [Full Token Allocation] * 20%) + [Star Lockdrop Participation Rate] * ([% of Stars Locked for 3 years] * [Full Token Allocation] * 40%) + [Star Lockdrop Participation Rate] * ([% of Stars Locked for 2 years] * [Full Token Allocation] * 60%) + [Star Lockdrop Participation Rate] * ([% of Stars Locked for 1 year] * [Full Token Allocation] * 80%)*

The same formula will apply to galaxies.

Five lockup periods are available upon claim. Assuming 100% of stars and galaxies are locked (ie no bonus is awarded), which is an implausible scenario, the allocation would be as follows:

Lockup Period (years)	**Maximum % Captured**	**Per Galaxy**	**Per Star**
1	20%	10,066	3,908
2	40%	20,133	7,816
3	60%	30,199	11,274
4	80%	40,265	15,632
5	100%	50,332	19,541

Any tokens left unclaimed due to either nonparticipation in the Lockdrop or participation in a lower than maximum duration Lockup will instead be allocated to those that participate in the maximum lockup period. This ensures that most tokens are awarded to those with the lowest time preference.

For illustrative purposes, using the above example, if 60% of stars were to lock five years and 40% for one year, the bonus pool accrued to five year stars would sum to 408,193,692 tokens for a pro rata bonus of 10,422 tokens per star (a 53% bonus).

Block Rewards and Transaction Fees

Zenith will issue block rewards over 32 years with four year halvings. Post-TGE, block rewards are the only mechanism by which new tokens will enter the network.

Block rewards will be received by validating Galaxies. Galaxies will distribute a portion of the reward to Stars that participate in proposing transactions. The specifics of the distribution between Galaxies and Stars are still under deliberation.

Transaction fees will eventually become the main source of revenue for Galaxies and Stars. We believe that 32 years should be ample time to establish Zenith as an independent chain, capable of generating enough demand for blockspace. This, in turn, will provide the incentive needed to maintain the chain's infrastructure.

Governance & Management

Zenith will begin development within a subsidiary inside of Tlon Corporation. Post testnet release, Zenith core development and business activity will be spun out into a Cayman Foundation.