CANOPY

An AI-Native Platform for Sovereign Blockchain Networks

Adam Liposky / Andrew Nguyen / Shawn Regan
adam / andrew / shawn @canopynetwork.org
Version 2.0 (June 2026)
Full Specification: github.com/canopy-network/canopy

Abstract.

The emergence of AI-assisted development tools has fundamentally altered the pace at which software is conceived, built, and iterated. Developers now move from idea to working prototype in hours rather than weeks, while iteration cycles that once consumed months have been compressed to days. Web3 infrastructure, however, was not designed for this velocity. Smart contracts remain constrained by host-chain languages and upgrade mechanisms, Layer 2 solutions inherit the technical and economic limitations of their underlying networks, and launching an independent Layer 1 requires significant engineering effort, validator coordination, and capital. Moreover, legacy Layer 1 codebases have grown so large and specialized that they exceed the practical context limits of modern AI development tools. Canopy is designed to close this gap by providing an AI-native development platform for sovereign blockchain networks. The platform is built on a simple thesis: *blockchains are applications, and applications are blockchains*. Rather than treating blockchain infrastructure as a scarce shared resource, Canopy enables every application to become its own fully programmable blockchain while retaining a clear path toward sovereign Layer 1 independence. Within this architecture, Canopy Network serves as the Security Root, staking collateral to secure Nested Chains: application-specific blockchains that receive consensus, peer-to-peer networking, and validator services through an operating system-level, plugin-based architecture. A novel restaking model eliminates the economic cold-start problem that has historically prevented independently launched blockchains from achieving meaningful decentralization without substantial prior fundraising. Nested Chains require approximately 200 lines of code to define, can be deployed in minutes rather than months, and are designed from inception to support AI-native development workflows. The platform consists of three integrated components. The Canopy Stack enables developers to build blockchains using AI-friendly languages including Go, TypeScript, Python, Kotlin, and C#, with scaffolding tools, semantic code indexing, and native integration with AI coding environments. The Canopy Terminal streamlines deployment, community bootstrapping, and token launch through either fiat payment or a bonding curve mechanism. By supporting the complete lifecycle of a blockchain, from development and deployment to shared security and eventual sovereign graduation, Canopy introduces a platform designed to move at the speed of AI while preserving the decentralization properties that make sovereign blockchain networks valuable.

Table of Contents

1. History

1.1 Bitcoin: The new paradigm

In 2009, the creation of Bitcoin [1] introduced a paradigm shift in the philosophy of technology, economics, and governance. Implemented as a peer-to-peer fully digital currency, Satoshi Nakamoto proposed a probabilistic solution to the Byzantine General's Problem called Proof of Work (PoW) that guaranteed the integrity of distributed data without depending on one central authority. By using cryptographic principles and the physical limitations of brute-force hashing, the miners of Bitcoin were able to validate transactions and append blocks to the blockchain in a secure and trustless manner. In doing so, Bitcoin was able to coordinate a world-wide digital payment system among permissionless participants. This idea of trustless infrastructure by way of decentralized technology began the proliferation of cryptocurrency and the blockchain industry as a whole. While Bitcoin's radical architecture and decentralization solidified it as the foundational blockchain, its design has fundamental limitations. By only optimizing for transfer and store of value, Bitcoin stopped short of enabling applications beyond currency. A lack of programmability, energy efficiency, and scalability highlighted the need for innovation beyond Bitcoin's foundational framework.

1.2 Ethereum: Programmable money

Four years after the invention of Bitcoin, Vitalik Buterin expanded the vision of Nakamoto by publishing a paper titled "Ethereum: A Next-Generation Smart Contract and Decentralized Application Platform" [2]. By evolving the blockchain state-machine from UTXO ledgers to multi-functional digital accounts, Ethereum enabled the creation of smart contracts, transforming blockchains from digital currency to dynamic ecosystems for generalized applications. This innovation introduced the idea of blockchain applications (dApps), allowing developers and enterprises to build automatic and decentralized services that leverage the advantages of blockchain technology such as censorship resistance and trustless coordination of permissionless actors.

Ethereum's innovation significantly increased its popularity, leading to an influx of transactions, resulting in congestion, high fees, and in some cases denial of service. While gas fees (financial penalties proportional to the computational weight of each transaction) were meant to serve as a mechanism to manage resource consumption, the marketplace design induced volatile and unpredictable costs which reduced its usability and introduced significant financial barriers. This phenomenon began years of research and development to scale Ethereum, underscoring the shortcomings of a unichain design in addressing the scalability and accessibility needs of blockchain applications.

1.3 Tendermint: A modular BFT blockchain

Shortly after Ethereum, Jae Kwon introduced Tendermint: "a byzantine fault tolerant state machine replicator for arbitrary deterministic, finite state machines" [3].

Similar to Ethereum, he sought to make blockchain technology accessible to the masses for developing and deploying generalized blockchain applications. However, Kwon's vision differed from Buterin by emphasizing a multichain model as the cornerstone of blockchain's future scalability and interoperability. To achieve this vision, Tendermint is not a standalone blockchain but rather an open-source, modular, consensus and peer-to-peer engine designed to streamline the development of L1 blockchains, by providing a foundational SDK for developers and enterprises to build upon.

In addition to its modular approach, Tendermint distinguished itself from Bitcoin and Ethereum by offering an energy-efficient consensus mechanism by adapting an existing solution to the Byzantine Generals problem, commonly known as Practical Byzantine Fault Tolerance. This, along with innovations by Peercoin [4] in 2012, resulted in the sybil protection mechanism known as Proof of Stake (PoS). With PoS, miners do not perform computationally intensive brute-force hashing; rather, they participate in stake-weighted voting, leading to them being reclassified as validators. Ethereum later adopted this concept, transitioning its network to PoS as a testament to its potential.

The creation of Tendermint marked a pivotal moment in blockchain history, as it made application-specific chain independence feasible. However, despite its potential, Tendermint did not achieve the level of popularity seen with Ethereum. Unlike Ethereum, Tendermint's design did not address ease of deployment and secure bootstrapping, which are essential for fostering widespread accessibility and mass adoption. Although Tendermint chains avoided the scalability constraints inherent to Ethereum smart contracts, without robust financial backing, an experienced development team, and skilled node operators, successful independent deployments of Tendermint became increasingly rare.

1.4 Cosmos: Introducing interoperability

In 2016, the Tendermint project evolved into "The Internet of Blockchains" when the Cosmos Network [5] and Cosmos SDK were revealed. The Cosmos Network was designed to provide a generalized communication framework called Inter-Blockchain Communication (IBC) to enable cross-chain functionality like the exchange of digital assets. The core product of Cosmos is facilitating multi-chain communication and data sharing between API-compatible sovereign networks through their dedicated Layer 1. This idea pioneered multi-chain interoperability, promising to bridge the gap between fragmented ecosystems while enabling scalability through a connected network of fully independent Layer 1 blockchains.

Though Cosmos and the IBC protocol were groundbreaking advancements, they faced several challenges that prevented them from being a comprehensive solution for developer mass appeal. The primary concern with the Cosmos design is that, much like Tendermint, it is simply a development framework to launch Layer 1s, enhanced with a standard communication protocol. While this approach certainly bolsters scalability, it suffers from many of the problems of Tendermint, not addressing bootstrapping challenges like lack of initial decentralization and economic security, and needing experienced network operators. Furthermore, while Tendermint and Cosmos offer a

modular template for consensus and peer-to-peer, users of the Cosmos SDK must build their own state-machine and governance mechanisms, as well as actively maintain the consensus and peer-to-peer modules they did not originally author. As a result, while Cosmos provided the foundational tools for building interoperable blockchains, it still lacked the bootstrapping framework necessary for widespread adoption and ease of use in the blockchain ecosystem.

1.5 Polkadot: Shared security as a service

In 2020, Polkadot [6], created by Ethereum co-founder Gavin Wood, presented a novel approach to blockchain scalability. Building upon lessons learned from Ethereum and Tendermint, Polkadot aspired to tackle the limitations of existing systems by offering an interoperable, multi-chain, shared security model. At its core, Polkadot maintains a PoS base-chain that connects to various sub-chains (termed "parachains" in the Polkadot ecosystem), enabling them to operate separately while using security from the validator set of the base-chain. Similar to Ethereum, the sub-chain's state machine is uploaded to the base-chain so validators may generically authenticate sub-chain blocks, but like Tendermint, the sub-chain maintains its own transaction and block database, lessening unichain transaction bloat issues. In addition to enhanced scalability, Polkadot's multi-chain system enables cross-chain interoperable communication and shared governance between sub-chains.

While the concept of Polkadot is innovative, the design has material flaws that have impacted its long-term adoption. Foremost, Polkadot achieves scalability through a paid service, meaning sub-chain access to Polkadot's security is limited and slots are auctioned periodically. This requires consumers to bond significant amounts of DOT tokens for extended periods, often causing liquidity constraints and creating high economic barriers to entry for sub-chains.

While Polkadot promises modularity of sub-chain operations, sub-chains are heavily dependent on the base-chain in a manner that is difficult to reverse. The design requires validators to execute sub-chain blocks using a WebAssembly State Machine (WASM) runtime binary on the base-chain. This forces each sub-chain to compile and upload its state machine to the base-chain (like a smart contract), necessitating programming language compatibility as well as careful coordination between the base-chain and sub-chains during any state-machine upgrades. Additionally, sub-chains rely on Polkadot's GRANDPA consensus mechanism for finality, meaning that if sub-chain independence were achieved (which is not currently supported), the finality of historical blocks secured by Polkadot would be threatened and potentially reversible. As with Ethereum, sub-chains are lifetime-dependent on Polkadot, and reversing this reliance would involve significant architectural changes or a complete deprecation of the chain's history.

Furthermore, the complex architecture of Polkadot imposes meaningful operational burdens. Navigating the multi-tiered relationships between the base-chain, the sub-chain, interoperable communication, and complex governance functionality often demands expertise that most development teams do not possess. Sub-chains are also required to elect dedicated participants who are responsible for aggregating transactions

and producing blocks, while other base-chain participants ensure their validation. The prerequisite of understanding and the burden of management create a level of complexity that can be overwhelming for new users and developers. Moreover, systemic reliance on Polkadot governance underscores design choices that result in ecosystem lock-in, presenting challenges in achieving complete decentralization and autonomy for sub-chains as they mature.

1.6 Rollups: The promise of scaling Ethereum

Around the launch of Polkadot, Rollups were popularized as the next-generation innovation for Ethereum scalability. In many ways, Rollups are similar to Polkadot sub-chains, offering a model that offloads heavy computation and state storage from the base-chain while retaining the security guarantees of the underlying protocol. Rollups achieve this by electing dedicated, often centralized, Sequencers who are responsible for aggregating transactions and producing blocks. However, Rollups are even more tightly coupled to the base-chain than Polkadot sub-chains, as they post critical transaction data to ensure data availability and maintain trust, especially given that Sequencers are typically centralized. The popularity of Rollups inspired projects like Celestia [7], led by Mustafa Al-Bassam, to champion data availability as an independent service, offering a Tendermint-based alternative for Rollup deployment. Unlike Ethereum Rollups or Polkadot's architecture, Celestia innovated by offloading transaction execution entirely to the sub-chains, further decentralizing computational responsibilities.

While Rollups promise scalability, they heavily detract from decentralization and introduce a complicated architecture. This type of implementation does not address ease of use nor graduation into fully independent systems, perpetuating concerns about scalability and long-term decentralization. Offloading block data to external protocols creates dependency and inefficiency, tying sub-chains to base-chains without a clear path to independence. Furthermore, data availability layers can exacerbate dependency concerns by requiring sub-chains to act as lite nodes to their own databases caused by the intermingling of all transactions on a shared layer. This design introduces inefficiencies and potential scalability issues for sub-chains as they require cryptographic proofs from the data availability layer to utilize their own data in a trustless fashion.

1.7 Avalanche: A marketplace of experienced validators

In 2020, Ava Labs introduced the Avalanche [8] protocol as a highly performant, multi-tiered blockchain model that enables the launch of customizable independent sub-chains (termed "subnets"). Avalanche positioned itself as a scalability leader by offering a Directed Acyclic Graph (DAG)-like consensus algorithm that enabled high throughput and low latency, allowing thousands of transactions per second. Unlike Polkadot, Avalanche does not attempt to provide shared security; rather, it serves as a platform for independent, API-compatible sub-chains to connect with a marketplace of credible validators to run their sovereign blockchain. To access the marketplace, the sub-chain developer typically stakes AVAX on the base-chain. This design prioritizes sub-chains maintaining their own security and sovereignty, employing sub-chains to attract base-chain validators through native token rewards. While validators are always

required to be staked on the base-chain, hired validators must typically stake on both the sub-chain and the base-chain layers, providing independent staking collateral for bad behavior on both levels. Taking further advantage of this model, Avalanche offers a native exchange platform, which enables token swaps between different sub-chains, creating a competitive advantage over less integrated ecosystems.

While there are many promising features of this protocol, some inherent challenges have contributed to relatively lower adoption. While the fundamental design of Avalanche focuses on sub-chain sovereignty, there are questions regarding the value proposition. The core architecture relies on sub-chains attracting and rewarding base-chain validators with the sub-chain's native asset. This raises questions about why sub-chains would use Avalanche: is any security improved from this design if the hired validators may switch allegiances for monetary value greater than the sub-chain's token reward? Furthermore, base-chain stake cannot be slashed by the sub-chains, blurring the lines between the appearance of safety and actually providing security.

Though API compatibility does lower the barrier for validators to support a sub-chain, Avalanche validators are typically required to stake on both the base-chain and the sub-chain. This raises concerns about how much Avalanche truly simplifies validator adoption, highlighting a lack of mechanisms to address financial barriers in the early stages of the blockchain lifecycle. Additionally, since the base-chain does not implement consensus or peer-to-peer for the sub-chain, it leaves developers responsible for creating their own consensus and networking layers, which is a high technical barrier to launch. Though the protocol does have the benefit of a native token exchange within the Avalanche ecosystem, it does not have much interoperable functionality outside of Avalanche-compatible sub-chains, presenting challenges in long-term sustainability and mass adoption.

1.8 EigenLayer: Restaking economics

Launched in 2023, a series of smart contracts on Ethereum called EigenLayer [9] introduced a novel economic model called restaking. Restaking allows Ethereum validators to reuse their bonded tokens to secure additional services and applications beyond just Ethereum's base layer. By leveraging Ethereum's consensus mechanism and validator set, EigenLayer enables decentralized projects to access an Ethereum-based, opt-in shared security model. Although this concept is similar to that of Polkadot and Rollups, the restaking economic model is innovative as it enables validators to secure sub-protocols using their already-bonded tokens by permitting the reuse of collateral. By doing this, a more capital-efficient architecture is produced, allowing validators to use their current stake to offer the service while lowering the cost for blockchain applications to access a shared security model. However, while EigenLayer's approach is useful for Ethereum dApps, it is not applicable to other systems because its restaking economics are exclusive to the Ethereum ecosystem. Furthermore, EigenLayer does not attempt to address interoperability, leaving these challenges to be solved by other technologies.

1.9 AI coding tools: A new development paradigm

Between 2021 and 2025, a generation of AI-assisted development tools fundamentally changed how software is written. GitHub Copilot, introduced in 2021, demonstrated that large language models could generate syntactically correct and contextually appropriate code from natural-language prompts at a fidelity sufficient for production use. Subsequent tools, including Cursor, OpenAI Codex, and Anthropic's Claude Code, extended this capability into full-workspace awareness: these systems can read an entire repository, understand its architecture, propose changes across multiple files simultaneously, and iterate on their own output in response to test failures or feedback. By 2025, surveys of professional developers consistently reported that AI coding assistants had reduced the time required to implement new features by an order of magnitude in domains where such tools were well-supported.

This acceleration exposed a structural asymmetry. In software domains with mature ecosystems, such as web development, mobile applications, and data engineering, AI tooling proved immediately productive because the underlying frameworks were designed with human-readable, modular, well-documented codebases that AI systems could parse and extend. Web3 infrastructure, by contrast, was not designed with these properties. The dominant paradigms, Solidity smart contracts, Rust-based execution environments, and large forked Layer 1 codebases, combine high complexity, immutability constraints, and opaque testing environments in ways that resist AI-driven iteration. A developer seeking to build a blockchain application in 2026 faces an acute version of the same bootstrapping problem that has impeded the industry for a decade, now compounded by the mismatch between the speed at which they can generate ideas and prototypes and the speed at which existing infrastructure can receive them. This gap is the central problem that Canopy is designed to close.

2. The Problem

2.1 Blockchain applications and the architecture of dependence

A blockchain application holds several critical advantages over traditional web applications by providing enhanced features that redefine how users interact with digital platforms. These properties are achieved by hosting the application on a network of peers (validators) that use a consensus mechanism to agree on the state of the application. The validators are able to interact trustlessly due to a predefined protocol that leverages cryptography for verifiability and integrity. Within this paper, the category of systems comprising dApps, Layer 2s, parachains, and side-chains are collectively referred to as Dependent Applications, to highlight their shared reliance on a base-chain for critical functions such as security, consensus, and governance. Their counterpart, the Layer 1 (L1), is a fully sovereign blockchain that implements its own consensus, peer-to-peer, and governance protocols without reliance on any external system.

One of the reasons Dependent Applications exist is that L1 blockchains are difficult to create. High economic value and a diverse set of validators are prerequisites to blockchain security. Due to the permissionless properties of a blockchain application, a low-value, weakly decentralized chain is susceptible to hostile takeover, commonly known as a 51% attack in proof-of-work networks or a 66% attack in BFT-based networks. Because of this vulnerability, most projects that achieved success with an L1 have required enormous fundraising events to personally guarantee the security of their blockchain from the outset. Additionally, an L1 system is generally considered difficult to develop. A Layer 1 blockchain typically consists of a predefined application (state machine) whose state is determined by validators via a consensus mechanism, communicated through a peer-to-peer network, and stored in a verifiable manner. Building such a system in an emerging industry can be considered challenging for developers who are unfamiliar with the complexities and nuances [10][11][12][13][14].

2.2 Smart contracts: Immutability as a barrier to iteration

Smart contracts on platforms like Ethereum are immutable by design. Once deployed, the code governing an application cannot be altered without migrating to a new contract address, a process that typically requires coordinating token migrations, updating user interfaces, re-auditing security properties, and managing community communication. This design choice, which was intended to provide guarantees of execution integrity, becomes a structural impediment in an era when AI-assisted development enables and encourages rapid iteration. A developer who can generate a new version of their application logic in an afternoon cannot deploy it without accepting weeks of migration overhead. The result is a category of blockchain application that cannot move at the pace of its creators.

Beyond immutability, smart contracts inherit the resource constraints and governance dependencies of their host protocol. Applications built on Ethereum are subject to Ethereum's block size, gas pricing, and validator censorship policies. In 2022

[15], the Ethereum community faced pressure from the United States government to comply with sanctions imposed by the Office of Foreign Assets Control (OFAC). At its peak, more than 50% of Ethereum blocks included only OFAC-compliant transactions, demonstrating a coordinated effort among validators to censor transactions from specific applications. Since Dependent Applications and Layer 2 solutions depend on a host protocol, they are ultimately non-autonomous and subject to censorship by parties who are not stakeholders in those applications. Furthermore, all Dependent Applications compete for the same block space, and high demand from a single application can cause network congestion and increased transaction costs for all others. This phenomena was illustrated memorably by CryptoKitties in 2017 [16][17], whose success effectively degraded the usability of the entire Ethereum network. Value generated by Dependent Applications accrues primarily to the host protocol in the form of gas fees, not to the application creating that value.

2.3 L2s and Rollups: Complexity without value capture

Layer 2 solutions and Rollup architectures were developed to address the throughput limitations of unichain models. They succeed, to a meaningful degree, in offloading transaction execution from the base-chain. However, they do so by introducing architectural complexity that works against developer accessibility. Rollups require language-specific execution environments that constrain what can be built and how. They impose their own sequencing and finality mechanisms that builders must understand and account for. They maintain a fundamental dependency on the base-chain for data availability and dispute resolution, making the Layer 2 subject to the same governance and regulatory exposure as the Layer 1 it inherits from.

Perhaps most importantly from a value creation perspective, L2 solutions systematically limit the economic capture available to builders. The value generated by activity on a Rollup flows predominantly upward to the base-chain and to the operators of centralized sequencers, not to the developers who built the applications creating that activity. An application that generates significant volume on an L2 is, in economic terms, a tenant paying rent to a landlord: the infrastructure benefits from the application's success far more than the application benefits from the infrastructure. The market has begun to price this dynamic explicitly. Hyperliquid, a single application-specific Layer 1 that retained full ownership of its execution environment and fee flows, achieved a fully diluted valuation of $57 billion as of the date of writing, exceeding the combined fully diluted valuation of the top 100 Layer 2 networks ($17 billion) [23]. The implication is unambiguous: when execution quality is high, the market rewards application-specific sovereignty over dependence on general-purpose settlement layers.

2.4 Legacy L1s: Codebases not designed for AI

Forking a legacy Layer 1 codebase, as pioneered by projects building on Cosmos SDK or forked Ethereum clients, represents the traditional path to application-specific chain sovereignty. This approach avoids the constraints of smart contracts and L2 architecture, but it substitutes one set of barriers for another. The codebases of mature Layer 1 protocols are large, highly specialized, and were written before AI coding tools

existed. They typically exceed hundreds of thousands of lines of code across multiple modules, written in low-level languages like Rust or Go with extensive use of domain-specific abstractions that AI systems trained on general programming corpora handle poorly. A developer attempting to use Claude, Copilot, or Cursor to extend a forked Cosmos SDK application chain will find that the AI frequently generates plausible-looking but functionally incorrect code, because the codebase is too large and too specialized for the AI to maintain a coherent working model.

Beyond the AI tooling problem, legacy L1 forks impose a security bootstrapping burden that is largely unchanged from the problems described in prior sections. A newly forked chain must independently recruit a validator set, bootstrap economic security from zero, and manage the full operational burden of a network before it has any users to justify that investment. The Cosmos ecosystem in particular demonstrated this barrier clearly: despite providing excellent tooling for building sovereign chains, the majority of projects built on the Cosmos SDK struggled to achieve meaningful validator decentralization and economic security without substantial prior fundraising. The technical accessibility of the framework did not eliminate the economic bootstrapping problem.

2.5 The dilemma summarized

The current state of Web3 development presents builders with two inadequate options. The first is to build a Dependent Application, accepting constraints on decentralization, value capture, scalability, and iteration speed in exchange for the convenience of existing infrastructure. The second is to launch a sovereign Layer 1, accepting the full burden of security bootstrapping, validator recruitment, codebase complexity, and operational overhead in exchange for sovereignty. Neither path is well-suited to a development environment where AI tooling enables rapid iteration and where application-specific chain sovereignty has demonstrably superior value capture dynamics. There currently does not exist a platform that bridges the gap between these two options while also being designed for the development velocity that AI tooling makes possible. Canopy is built to fill that gap.

3. The Canopy Developer Stack

Our flexible solution drives rapid development, dead-simple deployment, and unbounded growth. Together, the three components of the Canopy developer stack enable an application chain lifecycle that has not previously been possible. Builders develop in weeks, deploy in minutes, iterate rapidly, and retain the economic upside of their own network.

3.1 Canopy Stack: Build

The Canopy Stack is the development environment through which builders define and iterate on their blockchain application. Where legacy Layer 1 frameworks require deep familiarity with consensus internals, peer-to-peer networking, and cryptographic primitives before a developer can write a single line of application logic, the Canopy Stack abstracts all of that infrastructure and exposes only the application layer: the state machine that defines what the chain does.

A Nested Chain can be defined in approximately 200 lines of code. The framework supports five languages chosen specifically for their compatibility with AI coding tools: Go, TypeScript, Python, Kotlin, and C#. Each of these languages has extensive representation in the training data of major AI coding assistants, meaning that AI systems are effective at generating, debugging, and extending code written in them. This is not an incidental benefit; it is a design requirement. Canopy is built on the premise that the next generation of blockchain developers will build with AI as a primary collaborator, and the framework is architected to make that collaboration effective.

To further support AI-native development, the Canopy Stack provides context-assist scaffolding files, which are structured documentation of the chain's architecture that AI coding tools can use as context when generating code. A semantic code index with RAG (Retrieval-Augmented Generation) retrieval allows AI systems to locate relevant sections of the codebase without requiring the developer to manually identify dependencies. Repository-aware tooling, including ripgrep integration and AST (Abstract Syntax Tree) parsing, allows both developers and their AI assistants to understand the structure of the application at a semantic level rather than relying on text search. ML-driven iteration pathways enable safe upgrades without chain halts, addressing the immutability problem that makes smart contract development ill-suited to rapid iteration. The Stack integrates natively with AI coding environments including Claude, Codex, and Cursor, so developers can use their preferred tools without modification. Development time is reduced by an order of magnitude compared to forking a legacy Layer 1 codebase.

3.2 Canopy Terminal: Launch

Once a Nested Chain has been defined with the Canopy Stack, it is deployed through the Canopy Terminal. Where traditional blockchain deployments require weeks of validator onboarding, infrastructure provisioning, genesis configuration, and

community coordination before any users can interact with the network, the Terminal compresses this process to under five minutes.

The Terminal accepts deployment via a bonding curve launch mechanism. The bonding curve model is particularly significant: it enables community-bootstrapped deployment where the initial economic security of the chain is funded by community interest rather than by the development team's prior fundraising. This directly addresses the cold-start problem that has historically forced Layer 1 projects to raise substantial capital before launch. By replacing financial prerequisites with community popularity, the Terminal democratizes access to sovereign chain deployment.

The Terminal also provides virtual market momentum tracking and community activity tooling, giving builders real-time visibility into the adoption and economic health of their Nested Chain in the critical period following launch. The operational overhead that previously required a dedicated DevOps team is managed by the platform, allowing development teams to focus on their application rather than their infrastructure.

3.3 Canopy Chain: Grow

Nested Chains deployed through the Terminal receive Day-1 shared cryptoeconomic security from Canopy Network through the restaking and committee subsidization model described in detail in the Technical sections of this paper. This means that a chain deployed through the Canopy Terminal is not bootstrapping security from zero: it inherits the economic security of Canopy Network's validator set from the moment of its first block.

Composability with other Nested Chains within the Canopy ecosystem requires no bridges. The architecture enables trustless cross-chain interactions as a native feature of the platform rather than an afterthought requiring third-party infrastructure. Chains that reach sufficient maturity and economic security may graduate to sovereign Layer 1 status, at which point they preserve all historical finality through NestBFT's design (described in Section 8). The option to remain within the Canopy ecosystem indefinitely is also available; many applications will find that the shared security model continues to provide value long after they could theoretically support independent security.

The three components together define what it means for blockchains to be applications and applications to be blockchains. Canopy is a network of networks: a platform where each deployed chain is simultaneously a fully programmable application and a first-class participant in the broader ecosystem of sovereign networks. The vision for this platform, stated plainly, is millions of sovereign chains built by AI, deployed instantly, and upgraded continuously, where citizen developers build alongside professionals and infrastructure moves at the speed of ideas.

4. Technical Introduction

4.1 Preamble: The missing layer

An analysis of the advancement of blockchain technology in the past fifteen years demonstrates a narrow industry focus, one that has prioritized unichain throughput over the full application lifecycle. The trend has left certain trade-offs unresolved: supporting the complete lifecycle from bootstrapping to full sovereignty remains unaddressed; accessibility is hindered by complexity; and securing a blockchain requires substantial economic resources and operational expertise. The addition of AI-native development to this landscape has raised the stakes considerably. The absence of a platform that can receive rapidly iterated application code, deploy it with shared security, and evolve alongside it without requiring full rewrites is no longer a theoretical gap: it is a concrete barrier to the next generation of blockchain application development. Canopy addresses these challenges through a model that offers shared security services to blockchain applications with a seamless track to Layer 1 independence.

4.2 Introducing Canopy

Canopy is designed to launch and grow new blockchain ecosystems, enabling small applications (Nested Chains) to grow into fully sovereign Layer 1 networks. Unlike existing solutions that often cause lifelong, irreversible dependencies or require significant technical or financial resources, Canopy offers an accessible, capital-efficient alternative.

To accomplish this, Canopy offers a shared security service powered by a novel restaking economic model along with a fully programmable, modular framework for blockchain development. Through a novel consensus algorithm and unique design, the model introduces advanced mechanisms such as chain-halt rescue and long-range attack mitigation, enhancing the security of its Nested Chains. The platform's flexible plugin architecture enables one-way integration with projects outside of its ecosystem, enabling unprecedented ease of interoperability. The following sections provide a technical introduction to the protocol, covering the core concepts that enable Canopy to operate as a network of networks.

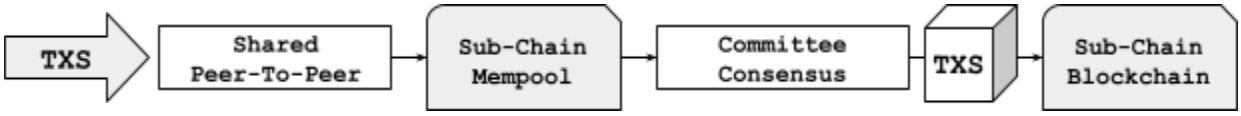
4.3 The protocol: An overview of key concepts

Canopy offers a unique combination of features that enable progressive blockchain ecosystem autonomy. Canopy provides blockchain developers consensus and peer-to-peer layers for Nested Chains through a dynamic set of Canopy Network validators. Validators stake the protocol's native cryptocurrency (CNPY) and are able to reuse the same collateral to validate multiple chains, improving capital efficiency. Canopy uses a plugin-based communication model, where validators interact with privately hosted Nested Chain nodes through bilateral APIs.

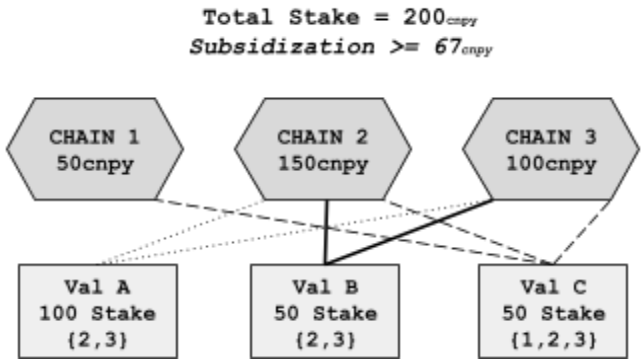
This design offloads state machine operation and block storage to the Nested Chain software, while managing the consensus and peer-to-peer layers of the process at the Canopy Network layer. Crucially, this architecture facilitates eventual Nested Chain graduation to independence with a predefined track, preserving the full finality of the chain's history through the mechanisms described in Section 8.

This unique plugin architecture extends beyond just new Nested Chains, enabling permissionless integrations to existing chains without needing modifications to their software or protocol. Validators using the same plugin are organized into committees, who work together to achieve Nested Chain consensus.

Unlike existing solutions, Nested Chain block production and transaction aggregation does not require any specialized actor. Instead, Nested Chain transactions flow through the shared network to their respective mempools, where a block producer aggregates them for consensus using NestBFT. After each Nested Chain block is committed, the block producer submits a summary transaction to Canopy Network, reporting information about what happened during the process, leading to things like reward distribution and stake slashes.



Canopy Network rewards follow a fixed inflation schedule and are divided equally among subsidized committees. A committee's subsidized status is based on the amount of validator tokens committed to each on Canopy Network. This means Nested Chains are able to have a dedicated committee if enough validators are willing to restake their collateral on their behalf, shifting requirements to fund blockchain security from financial investment from the project to community popularity.



Canopy also provides a pre-packaged state-machine software template for Nested Chains, complete with tools like a web wallet, blockchain explorer, and governance tooling for low-barrier, rapid development. Furthermore, while the platform's primary focus is to provide a framework for the lifecycle evolution of blockchain applications, it does not compromise on ecosystem interoperability, providing native token swaps and multi-tiered validator-driven governance.

5. Nested Chain Integration

5.1 Plugins: Modular integration

A plugin serves as a lightweight interface between Nested Chains and the validators of Canopy Network, enabling flexible construction and sovereign upgradability. Each plugin bridges both software via a simple API.

Algorithm 1 Plugin Interface

- `propose_block`: get txs from the sub-chain mempool to propose a block
- `validate_block`: validate a peer block by interpreting the txs
- `store_transaction`: saves a transaction to the sub-chain's mempool
- `store_block`: saves a block to the sub-chain block store
- `load_block`: loads a block from the sub-chain block store, enabling peer sync through the shared peer-to-peer network
- `start_bft`: begins the committee BFT instance, allowing the sub-chain to control its block times

By executing communication with a bilateral API and providing Nested Chains with an out-of-the-box template, the plugin enables Nested Chains to access a shared security model with minimal complexity. This approach combines the modular flexibility of Tendermint, without the insecurity of bootstrapping, and the shared security model of Polkadot, while avoiding its high complexity and rigid framework.

Critically, the plugin architecture is the property that makes Canopy compatible with AI-driven development workflows. When an AI coding assistant helps a developer iterate on their application state machine, the changes do not require coordination with Canopy Network validators, base-chain upgrades, or protocol-level governance votes. The Nested Chain upgrades its own software; the plugin interface remains stable. This sovereignty of upgradability is what enables the rapid iteration cycles that AI-assisted development demands.

5.2 Interoperability: External chains

Canopy's architecture is designed not only to integrate with new Nested Chains but also to plug into existing systems including legacy chains. This provides the ability to create generalized side-chain applications without requiring any modification to the existing chain software or protocol. The plugin model enables Canopy validators to observe and report on the state of external chains, validate activity, and distribute rewards based on that observation, all without the external chain needing to be aware of Canopy's existence. This design produces a passive, one-way integration model that offers a more realistic path to an interconnected network of networks.

Algorithm 2 QOS Validators for Tendermint Plugin (Example)

- The plugin notifies committee members of a new Tendermint block to start the consensus process.
- Each member reads the latest Tendermint block to identify which validators signed it.
- The committee reaches consensus on the list of signers.
- The committee then directs a portion of their CNPY block reward to the addresses that match the signers' public key on Tendermint.

This example demonstrates how Canopy's plugin architecture seamlessly integrates with existing blockchain systems to enhance their functionality and incentivize network participation. To further illustrate the point, below is a list of some applications that may be built by creating a plugin for an existing system:

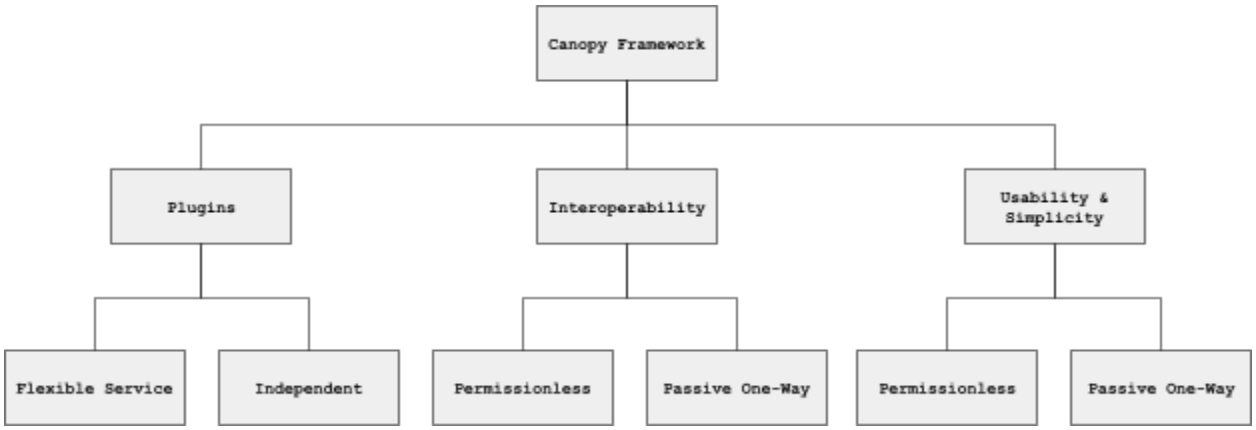
1. **Interoperability**
   - Cross-chain bridges
   - Native token swaps
   - Shared liquidity applications
   - Unified governance
2. **Scalability extensions**
   - Lightning network
   - Sharding network
   - Side-car dApp deployment
3. **Security extensions**
   - Chain-hault fallback
   - Dispute resolution framework
   - Checkpointing-as-a-service
4. **Governance extensions**
   - DAO coordination
   - Cross-chain voting
   - On-chain polling
5. **Decentralized oracle**
   - Chain state reporting
   - Off-chain data validation

5.3 Enhancements: Plugin-driven innovation

The plugin model offers greater flexibility than systems like Ethereum's Solidity framework by providing a more modular and customizable approach to decentralized application development. Unlike Ethereum, which permanently anchors a developer's implementation to the ecosystem, Canopy's design allows for a dynamic and programmable state machine template that is easily adapted to meet the utility of various applications. This architecture allows developers to deploy a wider range of decentralized applications without being lifetime-dependent on an ecosystem.

Similarly, Canopy's plugin framework offers greater usability and simplicity than Polkadot's WebAssembly State Machine (WASM) architecture. By requiring sub-chains to compile and upload their state machine to the base-chain, Polkadot's runtime design introduces deployment complexity and network coordination. In contrast, Canopy

implements a straightforward plugin architecture for cross-chain communication and consensus, ensuring sovereign upgrades without base-chain coordination. This not only simplifies the development but also reduces overhead, making Canopy the ideal solution for teams that would rather focus on their application logic than a coordinated plan for upgradability.



Additionally, Canopy's interoperability model surpasses Cosmos in ease of integration. Instead of requiring major protocol upgrades to achieve IBC compatibility, Canopy offers a passive, one-way integration model. This allows the external project to transition to, or be enhanced by, the Canopy platform without any protocol upgrades, offering a more realistic path to an interconnected network of networks.

6. Canopy Base-Chain

6.1 State-Machine: Accounts and validators

The foundation of Canopy's state-machine is an account-based, validator staking protocol. The native cryptocurrency (CNPY) is able to be transferred to and from accounts and surety-bonded by participants to become validators. When staking (bonding), validators are able to choose a variety of configurations. The more tokens the validator bonds, the more voting power it has in consensus operations. However, the more staked, the more at risk if slashed for bad behavior like double-signing or non-signing during consensus.

Stake Transaction

- **amount:** The amount of tokens to be removed from the sender account and locked as a surety bond against bad behavior.
- **committees:** The list of committees the validator is restaking their tokens towards.
- **is_delegate:** A non-delegate is registering for active participation in the committee consensus, whereas a delegate is passive participation that contributes its tokens to influence subsidized status.
- **is_compounding:** Automatically compounds rewards to the stake or early withdraws them to the reward address for a penalty.

Once registered, a validator is eligible to provide shared security services to any committee it chooses. In Canopy, Canopy Network functions as just another Nested Chain, leveraging the network's shared security. This design allows validators to secure other Nested Chains without being required to validate Canopy Network itself. Once in a committee, the validator connects with the other members of that committee through the shared peer-to-peer layer. Together the committee executes consensus for the Nested Chain, producing immediately final Nested Chain blocks. After each consensus round, the elected leader submits a summary transaction to Canopy Network, encapsulating the results of the round.

Summary Transaction

- **reward_recipients:** Who the committee agreed to distribute the block reward to.
- **slash_recipients:** Who the committee agreed to slash due to evidence of bad behavior.
- **orders:** Details of processed swaps for cross-chain liquidity.
- **checkpoint:** A verifiable sub-chain snapshot for Checkpoint-as-a-Service.

At any time a validator is able to update their stake information like committees and compounding status by submitting an edit_stake transaction.

Edit Stake Transaction

- **amount:** The updated amount of tokens being staked. This must be greater than or equal to the previous staked amount.
- **committees:** The update to the committees the validator is restaking their tokens for.
- **is_compounding:** The update to the auto-compounding status.

In addition to these transactions, the validator may also pause operations across all committees by submitting a pause transaction, temporarily removing it from service, and submit an unpause transaction to resume committee membership. To exit completely, the validator may submit an unstake transaction, permanently leaving the committees it is staked for. For short and long-range attack protection, the validator must wait for a governance-controlled number of blocks before their bond is fully returned, during which they are still eligible for slashing for bad behavior.

6.2 Token Swaps: Internal and external

Canopy is able to offer token swaps to any internal or external Nested Chain without requiring changes to the Nested Chain software integration. This is possible due to committees being trustless oracles for both Canopy Network and the Nested Chain. To begin the swap process, an asset seller may transfer funds to an escrow account controlled by the committee. This transaction includes metadata like an exchange rate for the Nested Chain asset and the seller's recipient address on the Nested Chain. The committee observes any acceptance of the ask on the Nested Chain through plugin-specific signaling like memoization that announces the buyer's receive address. Once accepted, the committee locks the escrow funds and awaits the buyer to transfer the Nested Chain asset to the seller's address. Once witnessed, the committee releases the escrowed funds to the buyer's address. It is important to note that memoization does not have to be in a memo field but can be signaled in any chain-specific way, like Bitcoin's Signature Script or Ethereum's Data field.

Canopy Token Swap Protocol

The Swap Protocol facilitates a token swap between Alice (with Token A) and Bob (with Token B). The committee oversees the process while controlling the escrow account of Blockchain A and observing Blockchain B.

1. Alice creates a *SellOrder* with the amount of *Token A* she wants to sell, the *exchange rate*, and her *Token B address*. Token A is escrowed in a committee-controlled address. Alice can reverse this order by submitting a transaction on Blockchain A.
2. Bob accepts Alice's offer by sending a transaction on Blockchain B, referencing Alice's offer hash and providing his *Token A address* in the memo field.
3. The committee updates the recipient of Alice's sell order to Bob's *Token A* address, verifying that Bob has enough *aged Token B* in his Blockchain B address.
4. Bob sends *Token B* to Alice, with a memo linking to the *Request to Sell*.
5. The committee witnesses Bob's transaction and releases Alice's *Token A* to Bob.
6. If Bob does not send *Token B* within N Blockchain B blocks, the committee resets Alice's *Request to Sell*.

6.3 Chain Halt Rescue: A unique value proposition

Due to the decoupled nature of consensus management and Nested Chain operation, the Canopy protocol is able to offer a unique value proposition: chain-halt recovery. Chain-halt typically occurs when there is a code error that leads to non-determinism in the state, preventing nodes from coming to consensus on the state of the ledger. Another common reason chain halts occur is a lack of a validator supermajority due to faulty validators. However, in Canopy, Canopy Network manages validator committees for the Nested Chains, ensuring that even if a Nested Chain encounters a critical bug or validator failure, the base-chain can reassign validators, restart consensus, or enable state rollbacks. This design emphasizes the robustness and fault-tolerant design of Canopy's architecture.

7. Consensus Background

7.1 Weak Subjectivity: The challenges of Proof of Stake

In 2014, Vitalik Buterin published a blog post [18] titled "Proof of Stake: How I Learned to Love Weak Subjectivity." In this paper, he describes how while PoS is an inherently more energy-efficient mechanism than its predecessor, it introduces the concept of a Long-Range Attack. An attacker executing a Long-Range Attack uses old validator keys, that once controlled a supermajority of the validator set, to rewrite the chain's history from a point far in the past. This is possible because the economic value that the keys once held has diminished and the attacker is presumably able to acquire them cheaply. Using these keys, an attacker is able to quickly rewrite the forward history of the blockchain to become the longest chain and potentially convince peers to join the fork, enabling economic attacks. In his post, Vitalik proposed the use of weak subjectivity, requiring node security to rely on social consensus by way of external checkpoints (finalized block hashes associated with specific heights) to mitigate this risk.

However, checkpointing as a mitigation for Long-Range Attacks has significant shortcomings. First, the design of constantly publishing and downloading checkpoints to validate a safe state is an administrative burden that may lead to participants ignoring the requirement, diminishing its effectiveness. More critically, the network security of a system is anchored to community-accepted checkpoint sources like software deployments and social media. This dependency heavily increases trust assumptions and makes the security of the blockchain fundamentally vulnerable to things like human error, political disputes, or coordinated attacks. This centralization not only introduces many points of failure but also poses a risk of censorship and manipulation, trading the Long-Range Attack vulnerability for many others.

7.2 Byzantine Fault Tolerance: HotStuffBFT

Byzantine Fault Tolerance (BFT) refers to a distributed system's ability to reach a safe agreement on some arbitrary data when a bounded number of participants act faulty or maliciously, addressing the classic Byzantine Generals Problem, where decentralized actors must coordinate to avoid catastrophic failure in the presence of traitors. Traditionally, BFT algorithms elect a leader among all distributed replicas to lead each view or round of consensus to completion. BFT is a fundamental prerequisite for decentralized networks and is the foundation of all modern blockchain consensus mechanisms.

In 2018, Cornell, UNC, and VMware research published a paper [19] detailing HotStuff BFT, "a leader-based Byzantine fault-tolerant replication protocol for the partially synchronous model." In this paper, the research team describes a highly optimized BFT implementation that has both only linear communication complexity and optimistic responsiveness. Over the past several years, HotStuff BFT has been rigorously peer-reviewed, offering academically proven safety and liveness guarantees in partially synchronous networks, with significant efficiency advantages over earlier

protocols like PBFT and Tendermint BFT. While this paper introduced exciting concepts, the absence of a practical pacemaker and leader selection mechanism has led to HotStuff BFT being regarded as a core academic reference rather than a production-ready implementation.

7.3 Algorand Consensus: Sortition leader election

In 2017, Algorand [20], pioneered by Silvio Micali and team, introduced a novel leader selection algorithm named Algorand Sortition. At its core, Sortition was designed to be a distributed denial-of-service (DDoS) attack-resistant solution, an area where competitors like Tendermint's round-robin algorithm fell short. The protocol works by each validator running a cryptographic lottery to determine their eligibility as a leader candidate. The lottery uses a cryptographic primitive called a verifiable random function (VRF), which guarantees unpredictable, uniform distribution without bias. Because the process is private, potential leaders are not known until they announce their candidacy, mitigating the risk of a DDoS attack on the leader. Algorand Sortition introduced a significant innovation in leader selection, inspiring implementations like Polkadot's BABE, enhancing both the security and scalability of blockchain consensus mechanisms.

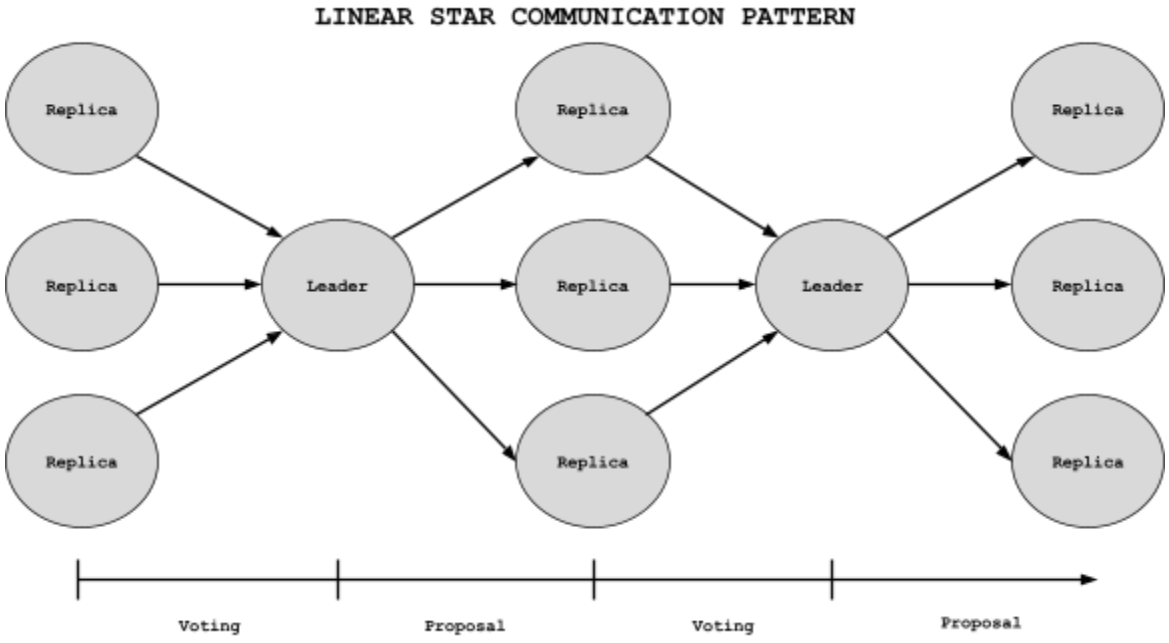
7.4 Verifiable Delay Functions: Chia Network

In 2021, Chia Network's [21] mainnet launched with a unique solution to Byzantine Faults called Proof of Space and Time (PoST). PoST utilizes a cryptographic primitive called a Verifiable Delay Function (VDF), which is designed to act as a reliable proxy of elapsed time. Unlike parallelizable hash computations used in Bitcoin's consensus algorithm, a VDF requires sequential computation, ensuring exponential advantage cannot be achieved with advanced or specialized hardware. Chia employs VDFs to create a predictable computational delay that ensures consistent block times and prevents bias in the consensus mechanism. In recent years, VDFs gained popularity as a useful cryptographic construct for blockchain technology, valued for their ability to introduce trustless time-based operations.

8. NestBFT Consensus

8.1 NestBFT: Fast, resilient and scalable consensus

Canopy presents NestBFT: an innovative consensus algorithm developed to (1) withstand grinding attacks while mitigating the risks of (2) DDoS and (3) long-range attacks. Unlike many modern consensus protocols that assume peer-to-peer network messages arrive without delay, this protocol is designed for unreliable peer-to-peer environments. NestBFT is engineered with a novel pacemaker mechanism to address validator coordination challenges, while offering immediate safety and finality. The protocol is optimized to be highly efficient, using BLS multisignature aggregation for O(1) space complexity while utilizing a star communication pattern to maintain linear communication complexity. This design aims to be intuitive and straightforward to implement, as Canopy employs NestBFT for both Canopy Network and Nested Chain layers.



8.2 Leader Election: Simple sortition and fallback

NestBFT implements a DDoS-resistant and highly available leader selection algorithm that is immune to grinding attacks. Drawing inspiration from Algorand Sortition, the protocol uses a simplified form of a Verifiable Random Function (a Practical VRF) in the form of a digital signature on seed data. The seed data consists of a fixed number of `last_proposer_addresses`, the current consensus `height` and `round`, ensuring unique input for each consensus view while being unable to be manipulated by a leader.

$$\text{Seed} = \text{Last\_Proposers\_Addresses} + \text{Height} + \text{Round}$$

$$\text{VRFOut} = \text{Hash}\left(\text{BLS\_Private\_Key.Sign}(\text{Seed})\right)$$

Each validator executes the VRF each consensus round to produce a unique `'VRFOut'`. Unlike many existing protocols that rely on manipulable seed inputs like the `last_block_hash`,, NestBFT's approach eliminates grindable bias, ensuring fairness and unpredictability. By combining simplicity with cryptographic integrity, this design enhances security and resilience in decentralized environments, offering robust protection against both DDoS and grinding attacks.

The next step of the election algorithm is for each validator to compute a numerical threshold that signals if a replica is a leader (block producer) candidate. This computation ensures stake amount proportionally weighs the chances of being the block producer. After this computation, the validator knows if they are a leader candidate by checking if their VRFOut is below the threshold.

$$\text{CandidateCutoff} = \frac{\text{voting power} \times \text{expected candidates}}{\text{totalVotingPower}}$$

$$\text{IsCandidate} = \text{toNumberBetween0And1}(\text{VRFOut}) < \text{CandidateCutoff}$$

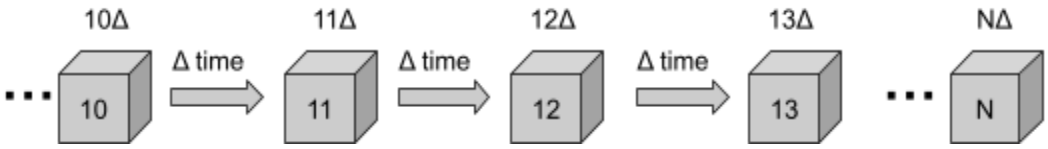
If deemed a candidate, the validator communicates the VRFOut to all replicas during the first Election consensus phase. Multiple candidates are expected and the validator with the smallest VRFOut is chosen as the leader. In the case of zero candidates, the leader election algorithm falls back to a simple stake-weighted random algorithm, which, while being less DDoS-resistant, is still immune to grinding attacks.

Algorithm 3 Fallback Leader Selection Algorithm

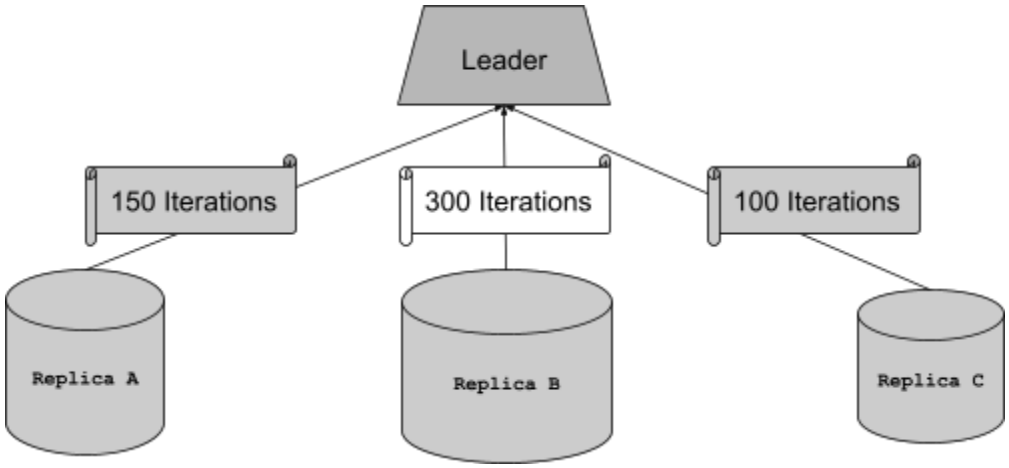
```
PowerIndex ← toNumber(Seed) mod validatorTotalTokens
index ← 0
for all Validator ∈ SortedVals do
    index ← index + Validator.StakedTokens
    if index > PowerIndex then
        return Validator
    end if
end for
```

8.3 VDFs: Verifiable mitigation of long-range attacks

Though the broader blockchain community accepts frequent checkpointing as a reasonable solution to prevent long-range attacks, NestBFT strives to further eliminate reliance on centralized sources and social consensus by offering a novel approach. Given that the VDF primitive is a reliable proxy for elapsed time and is fundamentally resistant to specialized hardware, NestBFT employs Verifiable Delay Functions to ensure that the addition of each block to the chain is temporally consistent. By requiring a sequential proof of elapsed time in each block, the recreation of the blockchain by a long-range attacker is infeasible, as it would require both significant computational resources and time to reconstruct a chain that is longer than the network.



Each validator replica runs a VDF for just under the expected block time with seed data from the most recent block. The VDFs are aggregated by the producer during the `ELECTION_VOTE` phase of NestBFT, which includes the VDF with the highest number of iterations in the proposal block. This design ensures the network gets the hardware benefit (if any) of the most computationally strong node in the network.



While this mechanism is effective in mitigating long-range attacks, the priority of Canopy is Nested Chain quality of life. By leveraging the long-range attack protection of Canopy Network, Canopy offers checkpointing-as-a-service to Nested Chains. Checkpointing information may be periodically included in a committee's summary transaction and is indexed separately for each Nested Chain, allowing full exportation of historical checkpoints upon sovereign graduation.

8.4 Phases: Step-by-step

This section explains the various phases of the consensus lifecycle, providing a thorough exploration of each stage to ensure a comprehensive understanding of how the consensus process works.

NestBFT Consensus Process

The consensus process of NestBFT is broken down into 8 core phases and 2 recovery phases. Each phase represents the smallest unit in the consensus process. Each round consists of multiple phases, and each height may consist of multiple rounds. These phases are executed sequentially to achieve consensus on the next block:

ELECTION

Each replica runs a Verifiable Random Function (VRF); if selected as a candidate, the replica sends its VRF output to the other replicas.

ELECTION-VOTE

Each replica sends ELECTION votes (signature) for the leader based on the lowest VRF value. If no candidates exist, the process falls back to a stake-weighted-pseudorandom selection.

PROPOSE

The leader collects ELECTION_VOTEs from +2/3 of the replicas, each including the lock, evidence, and signature from the sender. If a valid lock exists for the current height and meets Hotstuff's SAFE NODE PREDICATE, the leader uses that block as the proposal block. If no valid lock is found, the leader creates a new block to extend the blockchain. The leader then sends the new proposal (block, results, evidence) attaching the +2/3 signatures from ELECTION_VOTE to justify themselves as the Leader.

PROPOSE-VOTE

Each replica validates the PROPOSE message by verifying the aggregate signature, applying the proposal block against their state machine, and checking the header and results against what they produced. If valid, the replica sends a vote (signature) to the leader. Each vote vouches that the leader's proposal is valid.

PRECOMMIT

The leader collects PROPOSE_VOTEs from +2/3 of the replicas, each including a signature from the sender. The leader sends a PRECOMMIT message attaching +2/3 signatures from the PROPOSE_VOTE messages, justifying that +2/3 of the quorum believes the proposal is valid.

PRECOMMIT-VOTE

Each replica validates the PRECOMMIT message by verifying the aggregate signature. If valid, the replica sends a vote to the leader. Each vote vouches that the replica has seen evidence that +2/3 of the quorum believe the proposal is valid.

COMMIT

The leader collects PRECOMMIT_VOTEs from +2/3 of the replicas, each including a signature from the sender. The leader sends a COMMIT message attaching +2/3 signatures from the PRECOMMIT_VOTE messages, justifying that +2/3 of the quorum agree that a super-majority believes the proposal is valid.

COMMIT_PROCESS

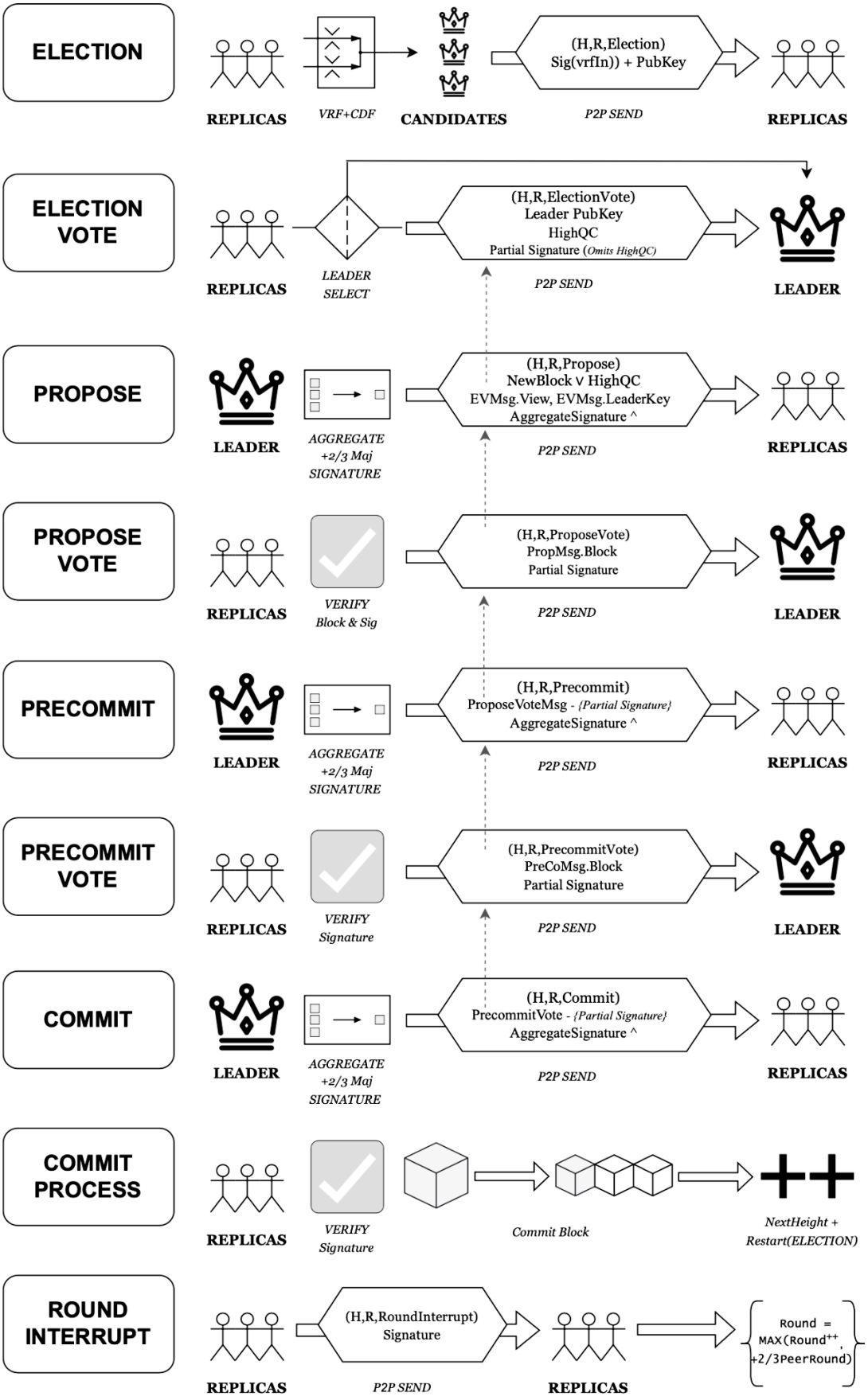
Each replica validates the COMMIT message by verifying the aggregate signature. If valid, the replica commits the block to finality, and resets the BFT for the next height.

ROUND_INTERRUPT (RECOVERY)

A failure in the BFT cycle caused a premature exit in the round. This results in a new round and an extended sleep time between phases to help alleviate any 'non-voter' issues. During this phase, each replica sends its View to all other replicas to alleviate round synchronization issues.

PACEMAKER (RECOVERY)

This phase follows ROUND_INTERRUPT. Each replica calculates the highest round a super-majority has seen and jumps to it to assist in 'round out of sync' issues.


ELECTION
REPLICAS
VRF+CDF
CANDIDATES
(H,R,Election)
Sig(vrfIn)) + PubKey
P2P SEND
REPLICAS
ELECTION VOTE
REPLICAS
LEADER SELECT
(H,R,ElectionVote)
Leader PubKey
HighQC
Partial Signature (Omits HighQC)
P2P SEND
LEADER
PROPOSE
LEADER
AGGREGATE +2/3 Maj SIGNATURE
(H,R,Propose)
NewBlock v HighQC
EVMsg.View, EVMsg.LeaderKey
AggregateSignature ^
P2P SEND
REPLICAS
PROPOSE VOTE
REPLICAS
VERIFY Block & Sig
(H,R,ProposeVote)
PropMsg.Block
Partial Signature
P2P SEND
LEADER
PRECOMMIT
LEADER
AGGREGATE +2/3 Maj SIGNATURE
(H,R,Precommit)
ProposeVoteMsg - {Partial Signature}
AggregateSignature ^
P2P SEND
REPLICAS
PRECOMMIT VOTE
REPLICAS
VERIFY Signature
(H,R,PrecommitVote)
PreCoMsg.Block
Partial Signature
P2P SEND
LEADER
COMMIT
LEADER
AGGREGATE +2/3 Maj SIGNATURE
(H,R,Commit)
PrecommitVote - {Partial Signature}
AggregateSignature ^
P2P SEND
REPLICAS
COMMIT PROCESS
REPLICAS
VERIFY Signature
Commit Block
NextHeight +
Restart(ELECTION)
ROUND INTERRUPT
REPLICAS
(H,R,RoundInterrupt)
Signature
P2P SEND
REPLICAS
Round =
MAX(Round++,
+2/3PeerRound)

8.5 Pacemaker: Optimal resync

NestBFT's novel pacemaker mechanism allows for optimal coordination in the case of an asynchronous networking event. Unlike legacy implementations like Tendermint, Canopy's pacemaker mechanism utilizes a communication phase to fast-forward to the highest round that a supermajority of replicas have seen. During the PACEMAKER communication phase, the replicas gossip their current round, allowing an efficient calculation of the highest round that a supermajority of validator stake have witnessed, which then triggers a transition to that round. The waiting time between synchronization phases increases linearly with each round, ensuring that the system adapts to the network's speed while preventing unnecessary delays. The phase wait time is calculated as: phase_wait = round * initial_phase_delta, allowing the system to gradually adjust to network conditions. The pacemaker combined with linearly increasing phase waiting time ensures a quick resynchronization of the validator set.

8.6 Preserving Finality: Life after independence

While applications have the option to stay on Canopy indefinitely, Nested Chains running on Canopy are designed to be able to graduate to independence without interruption. Unlike any existing technology, Canopy provides a seamless framework for independence transition.

Here's how it works:

Algorithm 4 Sub-Chain Graduation to Independence

1: **Input:** Current sub-chain state with validator set and blocks
2: **Output:** Transition to independent blockchain with continuity
3: Sub-ecosystem selects a transition block to switch to the stand-alone version of Canopy
4: Sub-chain plugin is configured to sign new validator set as NextValidators in the final incubator block header
5: Continuity of both the validator set and blockchain are maintained

If the Nested Chain is using the built-in checkpointing-as-a-service, it may easily export historical checkpoints and host them in a completely independent fashion. This ensures long-range attack protection by preserving the integrity and finality of the blockchain over time. Upon graduation, the Nested Chain retains full provenance of its history, including all blocks produced under Canopy's shared security model, with no requirement for chain migration, token swap, or re-genesis.

9. Tokenomics

9.1 CNPY: Overview

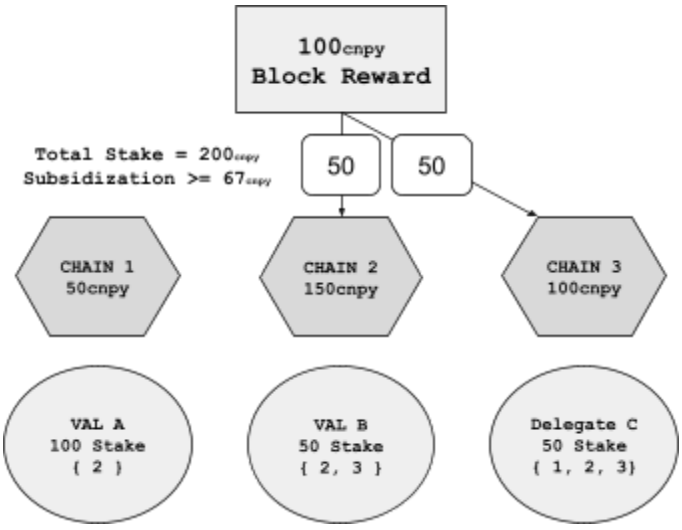
CNPY is the native token of Canopy, used for transactions and staked by validators to secure Canopy Nested Chains. Validators earn new CNPY tokens for providing security and consensus services, aiding the development of new blockchains. Users pay transaction fees in CNPY, based on computational needs and network load. CNPY has no pre-mint or pre-mine in the Genesis file, following managed fair-launch principles. Similar to Bitcoin, Canopy has a fixed total supply with halvings that occur in regular intervals throughout the lifespan of the project to limit the total supply and reward actors within the protocol.

9.2 Supply: Minting and limits

Every block produced on Canopy Network creates new CNPY tokens. The starting amount of the block reward is 80 CNPY. Blocks are created approximately every 20 seconds. Block rewards are halved every 3,150,000 blocks, or approximately every two years. The total tokens projected from block emissions are 504,000,000 CNPY.

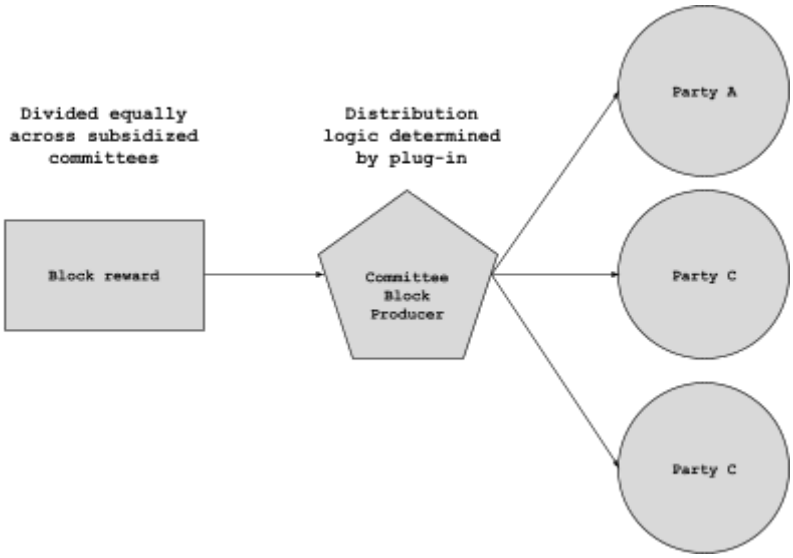
9.3 Block Reward: Fair distribution

The CNPY block reward is fairly distributed among network actors depending on their level of participation. Five percent of each block reward is allocated to the DAO Treasury Fund. The remaining block reward is divided evenly among all subsidized committees. Each committee distributes tokens according to its respective protocol, with the default distribution as follows: 70% to the CNPY Staker who serves as Block Producer, 10% to the CNPY Staker Delegate, 10% to the Native Token Staker Validator, and 10% to the Native Token Staker Delegate. Importantly, each block produced creates CNPY and Nested Chain tokens at the above distribution, resulting in a cross-pollination of tokens and actors that creates aligned economic incentives across the network.



A Nested Chain committee becomes subsidized once it surpasses a predefined threshold of total stake committed to it. Specifically, a committee must have more than 33% of the total stake committed to it in order to receive subsidization. This mechanism effectively makes community popularity the gating criterion for shared security access, rather than financial investment by the project itself.

The flexible reward distribution model enables numerous unique applications. For example, a plugin for an external chain could reward validators of that chain in CNPY tokens based on the quality of their service, incentivizing cross-chain collaboration and performance. It is worth noting that Canopy Network's own base-chain committee operates under the same funding model as other Nested Chains, making it architecturally indistinguishable from them.



9.4 DAO: The Treasury Fund

The DAO pool is a dedicated fund where 5% of the total block reward each block is set aside for project budgeting. The purpose of this treasury is to cover mandatory and discretionary spending that is for the good of the project, as determined by community sentiment through a supermajority agreement of validator representation. The DAO may allocate funds toward expenses such as exchange listings, launchpad partnerships, long-term development contracts, and ecosystem education. A supermajority of Canopy Network validators must agree to any transfer from the DAO Treasury.

9.5 Delegators: Influential, passive participation

Non-technical actors may participate in Canopy by being a Delegator: a participant who stakes CNPY but does not actively provide security services in committees. Delegators stake to receive part of the block reward of that particular chain. Importantly, Delegators have no slashing risk, which makes their role unique relative to active validators. Delegators play an important part in developer adoption, as they are the community-driven gatekeepers of Nested Chain subsidization status. Due to this,

delegators are likely to receive a significant percentage of rewards from each committee, as their contributions are necessary to retain native funding of the committee reward pool. Just like validators, delegators can restake their collateral toward multiple committees.

9.6 Restaking: Under the lens of Canopy

Restaking allows users to stake assets such as CNPY or other Canopy native assets across multiple Nested Chains, thereby extending Canopy Network's cryptoeconomic security to additional applications on the network. Effectively, restaking allows validators on Canopy to reuse their bonded collateral from one chain to secure multiple other chains, alleviating the cold-start problem that has historically required new Layer 1 networks to raise substantial capital before achieving meaningful decentralization.

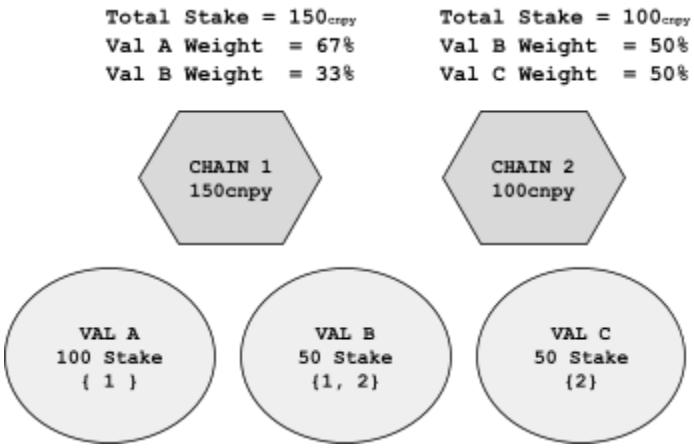
When a validator or delegate submits a stake transaction, they bond tokens from their address as a surety bond against bad behavior. In traditional staking, a validator's stake is only slashable for behaviors performed within the blockchain where the validator has locked its tokens. However, Canopy restaking allows the total reuse of staked collateral from Canopy Network to extend to multiple Nested Chains. This design enables stake to be used for security mechanisms and for subsidizing new projects, eliminating economic barriers to entry. This novel framework allows blockchain projects to launch with immediate economic security and have access to experienced node operators.

The utility of Canopy is to be the first Security Root for the ecosystem, seeding a peer-to-peer security future by sheltering Nested Chains. When validators perform consensus on behalf of subsidized Nested Chains, they earn CNPY as well as the native token of the Nested Chain. Validating and delegating in the Canopy protocol earns the participant a portfolio of tokens, not merely the speculation of value accrual but actual tokens from each secured chain.

Canopy's architecture promotes a self-reinforcing chain of Security Roots where mature Nested Chains shelter other new chains the same way their parent Security Root did for them. This means that the native assets of Canopy chains automatically have the same staking utility as their Security Root. Each asset earned in this portfolio of tokens enables participants to earn a new portfolio of tokens, a cycle that is exponential and continues from generation to generation, promoting compounding exposure to new assets.

$$\text{total\_committee\_val\_stake} = \sum (\text{validator\_stake\_committed\_for\_this\_committee})$$

$$\text{validator\_stake\_weight} = \frac{\text{validator\_stake}}{\text{total\_committee\_val\_stake}}$$



9.7 Auto-compounding: Usability and incentive alignment

As a means of reducing the workload for stakers, rewards are added to the stake of the validator or delegator automatically. If auto-compounding is off, an early withdrawal penalty is taken from the reward and burned, permanently removing those tokens from circulation. This feature both simplifies the staking process and encourages long-term participation and alignment with the network's growth.

9.8 Other incentives: The protocol after the block reward

Due to the halving mechanism, CNPY will eventually approach a terminal supply and native inflation incentives for committee participation will cease. However, the incentive structure can continue through various means. The following sections detail this point, underscoring the longevity of the system. This long-term vision takes great inspiration from Bitcoin's sustainability model, promoting an ecosystem where security is maintained through voluntary economic participation rather than perpetual inflation.

9.9 Nested Chain block reward: Aligning incentives

Unlike implementations like Avalanche, Canopy incentivizes committees to perform validation to ensure economic security is shared. However, Nested Chains are encouraged not to rely on native Canopy Network incentives alone. Much like traditional staking Layer 1s, a Canopy plugin may facilitate the distribution of a Nested Chain token directly to validators in the form of a block reward. This time-tested approach creates direct alignment between the Nested Chain and their committee. In order to remain competitive among other Nested Chains, it is incumbent upon each chain to determine appropriate additional incentives to encourage work and staking. This mechanism also creates long-term sustainability for Canopy. As Canopy Network block rewards decrease over time, these rewards can be replaced by Nested Chain rewards, slowly weaning the ecosystem off of newly minted CNPY. Eventually, CNPY will be solely used to provide security or discretionary incentives by projects.

9.10 Subsidies: Community-driven rewards

Chain supporters can choose to utilize the `SUBSIDIZE_TRANSACTION` on a particular Nested Chain to contribute CNPY to reward validators for their work. These subsidies are designed to create long-term alignment between the chain and their validators. Depending on the implementation, the subsidy eliminates the need for recurring payments, creates financial certainty for the stakers of a given Nested Chain, and incentivizes above and beyond the CNPY block reward. The subsidy transactions effectively create a two-sided marketplace where chains can pay validators for their block production. Mechanically, it transfers tokens to the committee's reward pool to be distributed as payments over a period of time set by the depositor. Using the OPCODE field of the transaction, subsidies are infinitely configurable to the needs of the chain.

Subsidies may be configured in various strategies. Because Canopy reads and analyzes Nested Chains, subsidies may be used to incentivize particular validators who meet certain criteria. This opens up opportunities for member chains to have increased control, flexibility, and creativity over validator rewards. As an example, a subsidy could be structured to be distributed only to validators that stake a certain amount of the chain's native token and meet certain quality-of-service requirements, creating a competition to stake more and maintain high performance. The maximum rewards to be distributed, the time range for which the subsidy is valid, and any potential strategies or multipliers must all be defined by the Nested Chain at subsidy configuration time.

10. Governance

10.1 Governance: The overview

Canopy is designed for effective on-chain governance and does not require the complexities that DAOs traditionally introduce. Canopy's governance structure utilizes validators as key decision makers in the system. Validators effectively act as elected representatives for Nested Chains, serving until such a time as the validator pauses or unstakes itself from the system. Validators, though not elected in the traditional sense, hold responsibility and authority by authorizing the inclusion of PARAMETER_CHANGE or DAO_TRANSFER transactions in blocks. Additionally, the platform includes built-in on-chain polling for any Nested Chain. This mechanism allows real-time straw polling of both token-holders and validators for any internal or external plugin, providing a transparent and accessible view of community sentiment and decisions.

10.2 Proposal Transactions: Parameter Change and DAO Transfer

Parameters are on-chain configurations that influence the behaviors of the blockchain state-machine, such as SEND_FEE or maximum BLOCK_SIZE. These structures are built-in controls that allow real-time modification of the base protocol without requiring a software upgrade or fork. To change the value of a parameter, a supermajority of committee members must agree on the new value. If no agreement is reached, the value remains unchanged.

As described in the Tokenomics section of this paper, the governance treasury account receives a portion of the block reward. The funds are discretionary and may be transferred to any address based on a supermajority agreement of committee members. This structure enables the DAO to capitalize community-driven initiatives and ecosystem development.

Governance Proposals and Transactions

1. Any persona may author a `PARAMETER_CHANGE_TRANSACTION` or `DAO_TRANSFER` using the built-in web wallet. These transactions have a start and end block window during which they are considered valid and may be submitted.

2. The author circulates the proposal throughout the community, communicating perspective and gathering support.

3. Validators may vote on this proposal using the web wallet, which populates a local JSON file in their data directory.
4. The author submits the transaction within the start and end block window.
5. Each validator processes the proposal according to its local JSON file, adding it to their mempool. A leader may then include the transaction in a block, with each replica voting based on their respective local files.
6. When the transaction is included in a final block, the proposal is immediately valid and applied at the same height.

10.3 Polling: Simple, adoptable, and extendable

Canopy enables Nested Chains to participate in a decentralized polling platform without embedding complex voting features into their main chain. This approach prevents storage bloat and ensures the scalability of the Nested Chain. Validator straw polling provides DAOs with real-time insight into voter sentiment on particular issues, enabling quicker decision-making. Additionally, Canopy's governance API collects voter metrics like voting power and actor type, helping Nested Chains understand community sentiment before formal voting takes place.

Polling through Canopy is designed to be simple and easily adopted. Validators can cast their poll-votes by including a specifically formatted memo in any transaction within the designated voting period. These memos are then captured through chain analysis and factored into the polling results, ensuring efficient and transparent governance without additional burden.

10.4 Cross-Chain Governance: Multi-chain DAO

Traditional Web3 governance is siloed by chain and project, leaving constituents unable to vote on the issues of the broader ecosystem. By interlinking validator committees across Nested Chains, Canopy lays the groundwork for a multi-chain DAO: a DAO of DAOs, an interconnected governance structure that merges interests between ecosystems and economies. As Nested Chains are intertwined through the Canopy protocol, an interconnected governance structure is born. With a DAO of DAOs, unprecedented governance stability, security, and multi-protocol cooperation is created, especially for newer and emerging chains.

11. Market Opportunity

11.1 App-chains versus L2s: The value capture argument

The market has already conducted a natural experiment on the question of whether application-specific sovereignty creates superior value relative to Layer 2 dependence. The evidence is unambiguous. Hyperliquid, a single application-specific Layer 1 that retained full control of its execution environment, fee economics, and governance, achieved a fully diluted valuation of $57 billion as of the date of writing. The top 100 Layer 2 networks, in aggregate, achieved a combined fully diluted valuation of $17 billion over the same period [23]. A single application-specific sovereign chain, built with focus and purpose around a specific use case, outvalued the entire category of Layer 2 infrastructure that currently represents the dominant paradigm of Web3 application deployment.

The mechanism behind this disparity is structural, not circumstantial. Application-specific chains capture the full economic value generated by activity on their network. Every transaction fee, every validator reward, every governance decision accrues to the stakeholders of that specific application and its native token. Layer 2 applications, by contrast, operate in an economic arrangement more analogous to tenancy: the infrastructure captures a significant fraction of the value generated by the applications it hosts, while those applications bear the constraints and risks of the underlying platform. As the Hyperliquid data point illustrates, the market assigns a meaningful premium to the property of economic sovereignty.

The thesis of Canopy is that Hyperliquid is not an outlier but a preview. The combination of AI-native development tooling, minute-scale deployment through the Terminal, and Day-1 shared security through Canopy Network's restaking model makes application-specific chain deployment accessible to a category of builder that has historically been excluded from this approach: teams without multi-year runways, without prior validator networks, and without the capacity to maintain hundreds of thousands of lines of infrastructure code. The opportunity is to create thousands of application-specific chains and capture value from the winners through CNPY's role as the shared security asset of the ecosystem.

11.2 The network of networks thesis

Canopy's architectural model creates value that compounds as the network grows. Each Nested Chain that joins the ecosystem adds to the validator committee network, deepens the restaking collateral base, and contributes its native token to the portfolio that validators and delegators earn. As Nested Chains mature, some will graduate to sovereign Layer 1 status and may themselves become Security Roots for new chains, creating a recursive, generational expansion of the network. The data moat effect, where AI models trained on the accumulated library of Canopy-deployed chains and templates become more effective at generating new chains, compounds the network's competitive position over time.

For exchanges and institutional actors evaluating the Canopy ecosystem, the relevant unit of analysis is not a single chain but the network as a whole. CNPY is the shared security asset of every Nested Chain in the ecosystem; as the number and quality of Nested Chains grows, the demand for CNPY as collateral grows with it. The restaking architecture ensures that this demand is not merely speculative but is grounded in the operational requirement of every committee, creating a structural connection between ecosystem growth and native token demand.

12. Concerns

12.1 Economic security with independence graduation

Since the Canopy model heavily promotes independence graduation, there is a natural inclination to believe that the long-term sustainability of Canopy Network may be threatened by the successful exiting of Nested Chains. Or that by prioritizing sovereignty, Canopy is unable to capitalize on the compounding value of the tokens within its ecosystem the way Ethereum has with its dApps. However, this logic does not consider the architectural implications of Canopy's tokenomics, which emphasizes Nested Chain value capture throughout the entire lifecycle of the application. Whether a chain is subsidized or not, it is incumbent on that application to provide native token incentive to its committee and delegators to remain competitive and subsidized throughout its usage. Moreover, the flexible reward distribution model allows for continued tokenomic interaction and support even after a Nested Chain graduates, allowing a sustained relationship with Canopy. Graduation from the platform does not necessarily mean a complete departure from the Canopy ecosystem: the Nested Chain remains API-compatible for plugin operations, maintaining the potential for ongoing collaboration. Furthermore, the validator set of the Nested Chain are likely committee members for other Nested Chains in Canopy as well, making full exits of committee members unlikely.

12.2 Centralization of validators and cascading failures

Because the Canopy model employs a shared validator set, there are concerns about increased centralization risks and the potential for cascading failures if one part of the system fails. The fear is that by having a single platform that supports the consensus and peer-to-peer layers for multiple Nested Chains, the architecture is more inherently fragile than legacy projects. However, a comparable analysis of existing systems demonstrates this concern to be unsubstantiated. For instance, Ethereum and Polkadot both utilize a single validator set to support their dependent chains or dApps; the difference is that the Canopy model does not enforce that every validator supports every Nested Chain, leading to a more resilient and flexible design.

To mitigate the possibility of cascading failures, the protocol implements a safety eject feature to prevent committees who overlap members from suffering a significant loss in security due to the byzantine behaviors of validators or Nested Chains. Canopy removes committee members who are slashed over a certain threshold from the committee, ensuring the protection of both parties from malicious or faulty actors. This mechanism safeguards the integrity of the network by isolating compromised participants before their actions can propagate further, eliminating real cascading risk.

12.3 Data Availability Layer

Popularized by the proliferation of Ethereum Rollups, data availability (DA) layer services are a prominent trend in blockchain infrastructure. DA layers abstract away the

storage of a sub-system's blockchain data to a base-chain protocol like Celestia, promising enhanced scalability and availability. However, the Canopy model outlines a novel abstraction that provides more flexible modularization to preserve the ledger while also being compatible with existing DA layers. Offloading or summarization of block data to another protocol cements a dependent relationship for the Nested Chain, causing further reliance on the base-chain system without a defined path to reversibility. With DA layers, Nested Chains effectively act as lite-nodes to their own blockchain database, which can be both inefficient during access and a barrier for independence. Furthermore, modularizing persistence to another system may actually threaten the scalability of the base-chain or be problematic for availability, as the maintainers of the data layer may not have natural incentives to preserve the ledger.

12.4 FPGAs and VDFs

For long-range attack mitigation, NestBFT relies on Verifiable Delay Functions acting as a reliable proxy for elapsed time. A fundamental property of this cryptographic primitive is resistance to specialized hardware and parallelization. However, research conducted in 2019 by VDFAlliance [22] uncovered a nontrivial speedup of the Wesolowski VDF using customized algorithms applied to Field Programmable Gate Array (FPGA) technology. VDFAlliance theorized a further speedup may be achieved using an application-specific integrated circuit (ASIC), but has not published updates since 2020. In order to mitigate this concern, NestBFT takes advantage of the fastest hardware on the Canopy Network committee. This means that even if only one committee member uses an FPGA or future ASIC, they would mitigate any hardware advantage an attacker may have. In addition, Canopy will rely on social consensus as a fallback, likely implementing checkpointing bi-annually, a significant improvement over legacy systems.

13. Conclusion

13.1 Canopy: A network of networks

The central insight of Canopy is simple in statement and consequential in implication: blockchains are applications, and applications are blockchains. The artificial distinction between infrastructure and application, between L1 and dApp, between platform and product, has constrained the blockchain industry to a design space that was defined before AI tooling existed, before the Hyperliquid data point demonstrated the value capture dynamics of application-specific sovereignty, and before the development velocity enabled by modern AI coding assistants made rapid chain iteration a practical possibility rather than a theoretical aspiration.

Canopy closes the gap between where the industry is and where it needs to go. The Canopy Stack makes blockchain application development accessible to any developer proficient in mainstream languages, with AI as a first-class collaborator. The Canopy Terminal removes the operational barriers that have historically gated chain deployment to well-capitalized teams with deep infrastructure experience. Canopy Network provides the cryptoeconomic security that makes a newly launched chain safe from Day 1, replacing the requirement for prior fundraising with community popularity as the gating criterion for shared security access.

In contrast to Avalanche and Cosmos, the Canopy protocol eases the required technical expertise and development challenges of running consensus and peer-to-peer for a Nested Chain through a shared set of experienced node runners. With NestBFT and the flexible plugin design, the model introduces novel security mechanisms for Nested Chains. The innovative restaking tokenomics creates unparalleled easy access for Nested Chains, shifting service accessibility from financial investment to community popularity. Unlike Ethereum and Polkadot, Canopy is the only design that addresses both bootstrapping and provides a predefined track to Layer 1 graduation, while providing frictionless upgrades during the application lifecycle. The unique plugin architecture enables unprecedented ease of interoperability, offering passive one-way integration for internal and external chains.

The long-term competitive position of Canopy rests on a compounding advantage: as more chains are deployed, the template library grows; as the template library grows, AI models trained on it become more effective; as those models become more effective, more chains are deployed. This cycle, combined with the generational recursion of the restaking architecture, in which mature Nested Chains become Security Roots for the next generation, positions Canopy not as a static protocol but as an expanding network that grows more capable and more defensible over time.

The vision remains: millions of sovereign chains built by AI, deployed instantly, upgraded continuously. Where citizen developers build alongside professionals, and infrastructure moves at the speed of ideas. Canopy makes this future inevitable.

References

[1] Nakamoto, Satoshi. "Bitcoin: A Peer-to-Peer Electronic Cash System." 2008. https://doi.org/10.2139/ssrn.3440802.

[2] Buterin, Vitalik. "Ethereum: A Next-Generation Smart Contract and Decentralized Application Platform." 2014. https://ethereum.org/content/whitepaper/whitepaper-pdf/Ethereum_Whitepaper_-_Buterin_2014.pdf.

[3] Kwon, Jae. "Tendermint: Consensus without Mining." 2014. https://tendermint.com/static/docs/tendermint.pdf. Archived: https://www.semanticscholar.org/paper/Tendermint-:-Consensus-without-Mining-Kwon/df62a45f50aac8890453b6991ea115e996c1646e.

[4] King, Sunny, and Scott Nadal. "PPCoin: Peer-to-Peer Crypto-Currency with Proof-of-Stake." 2012. https://www.peercoin.net/read/papers/peercoin-paper.pdf.

[5] Kwon, Jae, and Ethan Buchman. "Internet of Blockchains." Cosmos Network, 2016. https://v1.cosmos.network/resources/whitepaper.

[6] Wood, Gavin. "Polkadot: Vision for a Heterogeneous Multi-Chain Framework." 2016. https://polkadot.network/PolkaDotPaper.pdf.

[7] Al-Bassam, Mustafa. "LazyLedger: A Distributed Data Availability Ledger With Client-Side Smart Contracts." arXiv, 2019. https://arxiv.org/abs/1905.09274.

[8] "Avalanche Platform Whitepaper." Ava Labs, 2020. https://docs.avax.network/.

[9] Eigen Labs. "EigenLayer: The Restaking Collective." 2023. https://docs.eigenlayer.xyz/assets/files/EigenLayer_WhitePaper-88c47923ca0319870c611decd6e562ad.pdf.

[10] "Polkadot Passes the $140M Mark for Its Fund-Raise to Link Private and Public Blockchains." TechCrunch, 17 Oct. 2017. https://techcrunch.com/2017/10/17/polkadot-passes-the-140m-mark-for-its-fund-raise-to-link-private-and-public-blockchains/.

[11] "Blockchain Startup Raises a Quick $42M in First Sale." Cornell Chronicle, Aug. 2020. https://news.cornell.edu/stories/2020/08/blockchain-startup-raises-quick-42m-first-sale.

[12] "Multicoin Leads $20 Million Round for Speed-Focused Solana Blockchain." CoinDesk, 30 Jul. 2019. https://www.coindesk.com/markets/2019/07/30/multicoin-leads-20-million-round-for-speed-focused-solana-blockchain/.

[13] "Launching the Ether Sale." Ethereum Foundation Blog, 22 Jul. 2014. https://blog.ethereum.org/2014/07/22/launching-the-ether-sale.

[14] "How to Turn a $17 Million ICO Into $104 Million: The Cosmos Story." CoinDesk, 11 Nov. 2019. https://www.coindesk.com/markets/2019/11/11/how-to-turn-a-17-million-ico-into-104-million-the-cosmos-story/.

[15] Buterin, Vitalik. "Buterin Advocates for Censorship Tolerance in Special Cases." CryptoSlate, 7 Oct. 2022.

[16] ConsenSys. "The Inside Story of the CryptoKitties Congestion Crisis." ConsenSys, 18 Dec. 2017.

[17] Coin Metrics. "State of the Networks: Issue 281." Coin Metrics, 18 Nov. 2021.

[18] Buterin, Vitalik. "Proof of Stake: How I Learned to Love Weak Subjectivity." Ethereum Foundation Blog, 2014. https://blog.ethereum.org/2014/11/25/proof-stake-learned-love-weak-subjectivity.

[19] Abraham, Ittai, et al. "HotStuff: BFT Consensus in the Lens of Blockchain." arXiv, 2018. https://doi.org/10.48550/arXiv.1803.05069.

[20] Micali, Silvio, et al. "Algorand: Scaling Byzantine Agreements for Cryptocurrencies." arXiv, 2017. https://doi.org/10.48550/arXiv.1607.01341.

[21] "Chia Documentation." Chia Network, docs.chia.net/.

[22] Boneh, Dan, Joseph Bonneau, Benedikt Bünz, and Ben Fisch. "Verifiable Delay Functions." Stanford Applied Cryptography Group, 2018. https://eprint.iacr.org/2018/601.pdf. See also: vdfresearch.org.

[23] CoinGecko. Hyperliquid FDV and Top 100 L2 FDV Data. https://www.coingecko.com. Accessed at time of writing.