Flow

Blockchain Network Primer

Certain statements in this presentation constitute forward-looking statements. When used in this presentation, the words “may,” “will,” “should,” “project,” “anticipate,” “believe,” “estimate,” “intend,” “expect,” “continue,” and similar expressions or the negatives thereof are generally intended to identify forward-looking statements. Such forward-looking statements, including the intended actions and performance objectives of the Company and its affiliates involve known and unknown risks, uncertainties, and other important factors that could cause the actual results, performance, or achievements of the Company in its development of Flow, the blockchain, the network and the tokens as well as the features of Flow, the blockchain, the network and the tokens described herein to differ materially from any future results, performance, achievements, functionality of features expressed or implied by such forward-looking statements. No representation or warranty is made as to future performance or such forward-looking statements. All forward-looking statements in this presentation speak only as of the date this presentation was provided to you. The Company expressly disclaims any obligation or undertaking to disseminate any updates or revisions to any forward-looking statement contained herein to reflect any change in its expectation with regard thereto or any change in events, conditions, or circumstance on which any such statement is based.

Introduction

Flow is the blockchain for open worlds.

Flow is a fast, decentralized, and developer-friendly blockchain, designed as the foundation for a new generation of games, apps, and the digital assets that power them.

Flow empowers developers to build thriving crypto- and crypto-enabled businesses. Applications on Flow can keep consumers in control of their own data; create new kinds of digital assets tradable on open markets accessible from anywhere in the world; and build open economies owned by the users that help make them valuable.

Flow was developed with three key principles:

- **Developer-first experience:** Flow is designed for serious builders to get results fast. The architecture achieves 1000x improvements in speed and efficiency without breaking the network through sharding or "layer 2", preserving composability. Flow allows for snappy user experiences with deterministic finality. Upgradeable smart contracts and built-in logging support fit the requirements of modern software development while Cadence, a new programming language, makes writing smart contracts safe and easy.
- **Consumer-friendly onboarding:** Usage of crypto- and crypto-enabled applications is held back by the high-friction experience of accessing current networks. Flow was designed for consumer engagement from day one, with mainstream-ready payment onramps allowing a safe and low-friction conversion from fiat to crypto for most consumers. Flow also incorporates protocol-level usability improvements such as smart user accounts and human-readable security, setting a strong foundation for adoption.
- **Mainstream-ready apps and distribution:** Awesome content is the best catalyst for adoption, jump-starting consumer interest and kickstarting a flywheel with developers. Flow has global reach from day one, with committed partners representing leading technology firms as well as some of the most-loved brands in the world. The full composability possible on Flow allows developers to safely and easily build on top of each others' code, creating entirely new products and services at an accelerating pace. Composability allows developers to innovate faster, leading to more consumer choice.

Importantly, Flow is also securely decentralized. The Flow architecture supports large numbers of participants with a range of technical and financial commitments, resulting in a system that is easy to join while being difficult to subvert.

Developer-First Experience

Our experience developing blockchain applications like CryptoKitties and the Dapper Smart Contract wallet has led us to incorporate a number of usability improvements into the protocol layer on Flow. Several are outlined below.

Upgradable Smart Contracts

One of the most important promises made by smart contract platforms is that users can trust the smart contract code instead of trusting the smart contract authors. This aspect of blockchains unlocks use cases that we are only beginning to explore, the most impactful of which might be the concept of open services and composability.

In their first incarnation, smart contract platforms were designed such that contract code could never be changed after it is released. This is the most straightforward method to achieve the goal: If the code can't be changed, even by the original authors, you clearly don't need to trust the authors after the code is launched.

Unfortunately, software is hard to get right the first time. There are no shortage of examples of smart contracts that – even with incredibly talented teams and motivated communities – had subtle problems that led to a massive loss of funds.

Many developers have expressed the desire to fix or improve a smart contract after it has been deployed, and several have gone to a lot of time and trouble to build some mechanism into their smart contract to allow for upgrades or migrations. But having each developer "roll their own" mechanism for upgradability adds complexity, and makes those smart contracts harder to trust.

On Flow, we allow smart contracts to be deployed to the mainnet in a "beta state", where the code can be incrementally updated by the original authors. Users will be alerted to the unfinished nature of this code, and can choose to wait until the code is finalized before trusting it. Once authors are confident that their code is safe, they can irrevocably release their control on the contract, and it becomes perfectly immutable for the rest of time.

This system balances the needs of users to be informed about what kind of code they are dealing with – whether or not an application or smart contract is truly trustless – while allowing developers the flexibility to tweak their code for a limited time after shipping.

Fast, Deterministic Finality

From the standpoint of end users, the speed of a blockchain is most practically measured by the time it takes before they (or their client software) can be confident their transaction is permanently included in the chain. This is commonly referred to as “finality”. In Bitcoin, most people define finality as six block confirmations which can take more than an hour. Ethereum improves on this by achieving probabilistic finality after about 6 minutes.

On Flow, deterministic finality is achieved within seconds: once Consensus Nodes determine which block a transaction will be a part of, user agents can in most cases execute the transaction locally and give feedback to the user almost immediately. In cases where results may be influenced by other transactions in the network, users will either choose to trust an execution node, using modern APIs to get feedback within a couple of seconds, or wait until the results of the transaction are sealed into the blockchain along with all the relevant execution and verification receipts. This process of block sealing and formal observation takes around 10 blocks; about ten seconds at launch.

Built-in Logging Support

Sometimes, the only way to be sure that a complicated piece of software is working as expected is to log its behaviour, in detail, over a long period of time. Existing smart contract platforms don’t include a logging facility for the simple fact that storing a complete log record of the entire blockchain is completely intractable. Too much data!

Flow recognizes that since all smart contract transactions are fully deterministic, there is no need to store the actual logs for every transaction inside the network. Instead, Flow simply marks which transactions would have produced log messages for which topics. If someone wants to “examine” the logs, they can query the blockchain for the subset of transactions tagged with a particular topic and then re-run the transactions locally to generate those logs for analysis. This technique also makes event logging dramatically more efficient.

Cadence

Cadence is a new programming language designed specifically for the new paradigm of crypto-enabled applications. Compared to existing languages, Cadence introduces features to smart contract programming that help developers ensure that their code is safe, secure,

and approachable. Some of these features include first-class resources and a strong static type system as well as built-in pre- and post-conditions for functions and transactions.

Consumer-Friendly Onboarding

Human Readable Security

On current networks, it's nearly impossible for an app or wallet software to provide a human-readable message clearly outlining what permissions they're giving when authorizing a transaction.

The Flow transaction format makes very strong guarantees about what kinds of changes a transaction can and can not make. This makes it easy for the wallet to ensure users are making informed decisions about what they are approving. For example, "This transaction can transfer Kitty #1693842, but it can't access any of your other tokens."

It will be up to wallet software to display this information to the users, but by making the Flow transaction format easy to statically analyze, we create the possibility for a more transparent transaction approval process.

Smart User Accounts: no more seed words or lost keys

The Flow account model is designed with flexibility in mind. Over the past year, Dapper Labs has pioneered a variety of usability enhancements to the Ethereum account model as part of the Dapper Smart Contract Wallet. Those enhancements, and more, will be part of the native account model on Flow:

- User agents can easily make sure consumer never lose their assets – or access to their accounts – without relying on complex security
- Added security through optional multiple signature support, with the ability to cycle out old keys regularly to avoid security leaks
- Seamless and safe user experiences for consumer apps and other trusted entities with easily delegated authority for specific actions
- Optional, modular, smart contract functionality built into every wallet at the protocol level to support automated processes or more sophisticated authorization controls

Technical Architecture

In a traditional blockchain, every node stores the entire state (account balances, smart contract code, etc.) and performs all of the work associated with processing every transaction in the chain. This is analogous to having a single worker build an entire car.

From manufacturing to CPU design, **pipelining** is a common technique for dramatically scaling up productivity.

Flow applies pipelining to blockchains by separating the jobs of a cryptocurrency miner or validator into four different roles: Collection, Consensus, Execution, and Verification. This separation of labor between nodes is vertical (across the different validation stages for each transaction) rather than horizontal (across different transactions, as with sharding).

In other words, every validator node still participates in the validation of every transaction, but they do so only at one of the stages of validation. They can therefore specialize for — and greatly increase the efficiency of — their particular stage of focus.

This means that smart contracts and user accounts on Flow can always interact with each other in one atomic, consistent, isolated, and durable (ACID) transaction. In other words: all dapps on Flow run in the same shared execution state. This ensures good user experience and full composability, letting developers easily build on each other's work.

Problems with Sharding

Most decentralized systems propose to improve throughput by fragmenting the blockchain into a series of smaller, inter-connected networks forced to communicate asynchronously. This approach comes under a variety of different names (sharding, side-chains, etc.), but the essence remains the same: break the blockchain's state into independent pieces.

This fragmented state removes the ACID properties from transactions in the network. Loss of ACID guarantees makes building an app that needs to access data across fragments far more difficult and error-prone.

Sharding effectively saddles the hardest part of scaling the blockchain onto application developers rather than solving it at the protocol level.

Sharding isn't typically a problem for simple token transfers, but even simple interactions between smart contracts become very complicated in a sharded environment. This dramatically limits the composability and therefore network effects of smart contracts.

Separating Consensus from Compute

Tasks within a blockchain can be divided into two types:

- Non-deterministic (or "subjective") tasks, such as determining the presence and order of transactions in the blockchain
- Deterministic (or "objective") tasks, such as computing the result of those ordered transactions once it has been determined

Non-deterministic tasks require a coordinated consensus process (like Proof of Work or Proof of Stake). Deterministic tasks, on the other hand, always have a single, objectively-correct outcome. The critical insight behind Flow's architecture was that the single biggest bottleneck to blockchain performance is the deterministic task of executing transactions after they've already been included into a block, and not the subjective process that requires consensus, i.e. the formation of the block itself.

The Flow architecture separates non-deterministic processes from deterministic ones and assigns each to different types of nodes based on their technical capabilities.

Multi-Role Validator Nodes

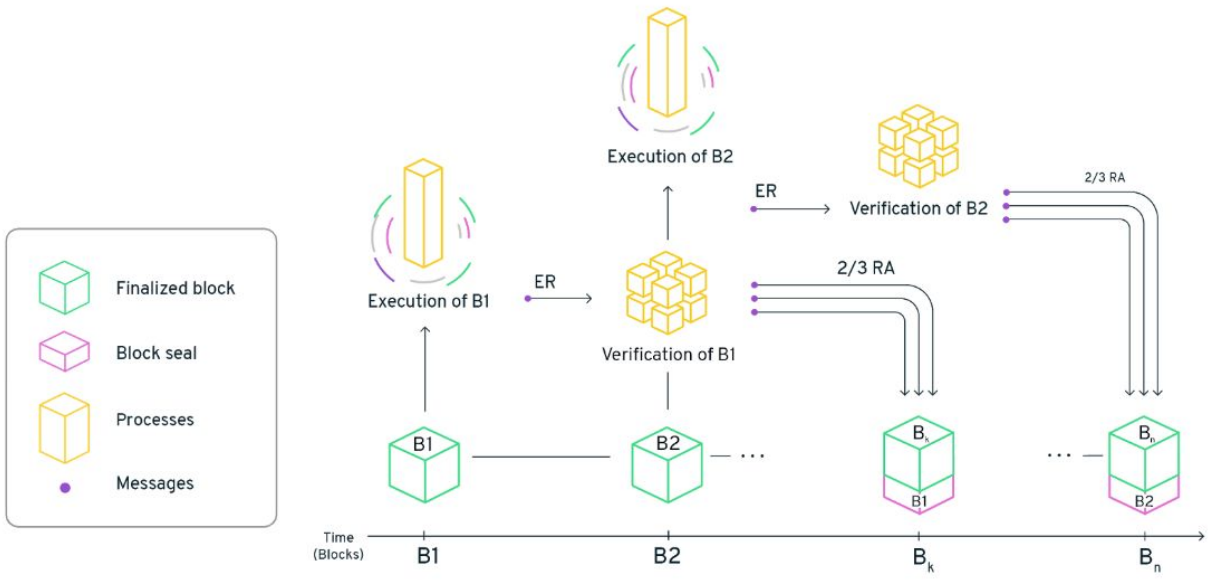
Flow pipelines the work of a blockchain miner or validator across four different roles that require staking; a separation of concerns that significantly reduces redundant effort:

- Consensus Nodes decide the presence and order of transactions on the blockchain
- Verification Nodes are responsible for keeping the Execution Nodes in check
- Execution Nodes perform the computation associated with each transaction
- Collection Nodes enhance network connectivity and data availability for dapps

Consensus and Verification Nodes together are the foundation of security in the Flow network. They leverage cryptoeconomic incentives to hold the rest of the network accountable. These validators can optimize for security and decentralization. Execution and Collection Nodes, on the other hand, do work that is fully deterministic – making them less vulnerable to attack. The work of these nodes is also verified and held accountable by the other node types. Execution and Collection Nodes can therefore safely optimize for security and scalability, allowing the network to scale. Operating these nodes requires dedicated

server hardware in a professionally managed data center.

Flow is designed such that even a single honest node, of any role, can find, punish, and trigger recovery from invalid data introduced by dishonest Collection or Execution Nodes.



The roles of Consensus and Verification are streamlined to allow high levels of participation, even by individuals with consumer-grade hardware running on home internet connections.

Participation as a Consensus Node is constrained by the limitations of the consensus algorithm, but easy bonded delegation at the protocol level will let any token-holder participate in the Flow consensus process. Flow is currently being developed using a variant of HotStuff, one of the most secure and well-understood proof of stake algorithms, but can easily adapt to new consensus approaches as the state-of-the-art moves forward.

Specialized Proofs of Confidential Knowledge (SPoCKs)

Specialized Proofs of Confidential Knowledge (SPoCKs) are a new cryptographic technique developed by the Flow team, formally defined in our Technical Papers. SPoCKs allow any number of provers to demonstrate to a third-party observer that they each have access to the same *confidential knowledge*. These proofs are non-interactive and don't leak any information about the confidential knowledge itself. Each prover's SPoCK is *specialized* to them, and can't be copied or forged by any other prover. Flow uses SPoCKs to address the Verifier's Dilemma by requiring Execution and Verification Nodes to "show their work". In

order to get paid, these nodes need to provide a SPoCK showing access to confidential knowledge that can only be obtained by executing all of the transactions assigned to them.

Flow Community

Flow is committed to a world of open ecosystems: a world where software developers, content creators, and consumers alike are appropriately incentivized and rewarded for the value they contribute to the network.

The technical architecture is just one example of how Flow will ensure inclusivity and participation at the protocol level: our community evangelism and efforts toward governance are equally if not more important.

Node Operator Fees and Rewards

On Flow, validator node operators supporting the network receive a portion of the transaction fees that pass through the system proportional with the work they do and their associated stake. Unlike miners or validators for other blockchains, validators on Flow can get started as Consensus or Verification Nodes with relatively cheap hardware, ensuring broad, equitable participation and decentralization.

In the early years of the network when fees are low, the network will provide additional rewards to node operators proportional with their efforts and associated stake.

Developer Ecosystem

A healthy and vibrant ecosystem is the most important long-term determinant of success for a blockchain. It is a fundamental requirement that Flow engages with a large and diverse set of stakeholders. Most importantly, this extends beyond the investor base and includes the developers and ecosystem partners that choose to build on top of the network.

In addition to a technical design optimized for developer experience and performance, the Flow team is taking additional steps to ensure a healthy ecosystem:

- **Developer Early Bird Program:** over the coming months, the Flow team will begin demonstrating the capabilities of the network to interested blockchain developers for

technical feedback. Whether you're an independent hobbyist or a venture-backed powerhouse, the Flow team wants to hear from you.

- **Ecosystem Development:** a portion of Flow tokens will be set aside for ecosystem development to bootstrap adoption and reward early participants in the network. These participation rewards will be distributed via a number of different programs including competitions, hackathons, and contributions to open source development. In addition to accelerating adoption, setting aside a portion of tokens for long-term ecosystem development also ensures a path to diversifying and decentralizing network participation and governance, ensuring global access from a variety of participants.

Content Partners

Blockchain lets brands and influencers connect directly with their fans in new ways. Features like digital scarcity and true ownership of assets create the space for gamified social experiences that go beyond individual apps, creating collector economies around every unique IP. Flow is working with independent developers that are breaking the mold as well as some of the world's leading entertainment studios, IP holders, and publishers to ensure our platform serves their needs.

Flow will sponsor the creation of an entertainment industry council for C-level executives from global IP holders, game publishers, entertainment studios, and cultural influencers. The council will help identify risks and opportunities, remove friction-points for consumer adoption, and help catalyze a healthy global entertainment ecosystem.

Get started by joining the community

Together with our community Flow can power an open and trustworthy internet for billions of consumers. We're looking for developers, companies, and ambassadors to bring this new digital world to life. These teams and individuals will work closely with our team and receive mentorship in order to establish sustainable products, services, businesses, and communities on Flow. If you're interested in building the future on Flow, please tell us more.

What's Next

Tell us about your project

Read the Flow Technical Papers

Follow us on Twitter

Join our Discord

Further Technical Reading

The architecture of Flow is specified in a series of Technical Papers which explain the many nuances of the system. Technical Papers 1 & 3 have been released first because they are the most critical to understand and assess the soundness of the system while also including the results most applicable to other projects.

Technical Paper 1: Separating Consensus & Compute

The first paper describes the approach at the core of the Flow architecture: splitting consensus (selection and ordering of transactions) from compute (executing each transaction and recording its output) and proves this can dramatically increase throughput without compromising security. In this first paper we analyze how the Flow architecture increases performance, preserves ACID guarantees, and prove that it does not compromise security. The result is a throughput increase by a factor of 56 compared to conventional architectures without loss of safety or decentralization. The paper also notes that a working system based on these ideas must verify the computation (the subject of technical paper 3), but that its key result is applicable regardless of how that problem is addressed.

Technical Paper 2: Block Formation

The second in the series of technical papers formalises the process of block formation and the Proof of Stake based consensus process in Flow. There is relatively little “uncharted territory” in this area of the protocol as Flow simply adapts a variant of the HotStuff consensus algorithm for Consensus Nodes to come to consensus on the blocks they will honor at every block height. This paper also addresses the Consensus role’s responsibility in mitigating challenges submitted to the network.

Technical Paper 3: Execution Verification

The third technical paper answers questions posed by the first whitepaper around the verification of computation results. The paper provides a formalization of our verifiable

computation scheme with proofs of safety and liveness under reasonable Byzantine assumptions. Although the paper does not explore the possibility, we believe this result could be adapted to other scenarios where Bulletproofs, TrueBit, TEEs, and other verifiable computation schemes are applicable.