Avalanche Platform
2020/06/30

Kevin Sekniqi, Daniel Laine, Stephen Buttolph, and Emin Gün Sirer

Abstract. This paper provides an architectural overview of the first release of the Avalanche platform, codenamed Avalanche Borealis. For details on the economics of the native token, labeled $AVAX, we guide the reader to the accompanying token dynamics paper [2].
Disclosure: The information described in this paper is preliminary and subject to change at any time. Furthermore, this paper may contain "forward-looking statements."[1]

Git Commit: 7497e4a4ba0a1ea2dc2a111bc6deefbf3023708e

1 Introduction

This paper provides an architectural overview of the Avalanche platform. The key focus is on the three key differentiators of the platform: the engine, the architectural model, and the governance mechanism.

1.1 Avalanche Goals and Principles

Avalanche is a high-performance, scalable, customizable, and secure blockchain platform. It targets three broad use cases:

- Building application-specific blockchains, spanning permissioned (private) and permissionless (public) deployments.
- Building and launching highly scalable and decentralized applications (Dapps).
- Building arbitrarily complex digital assets with custom rules, covenants, and riders (smart assets).

[1] Forward-looking statements generally relate to future events or our future performance. This includes, but is not limited to, Avalanche's projected performance; the expected development of its business and projects; execution of its vision and growth strategy; and completion of projects that are currently underway, in development or otherwise under consideration. Forward-looking statements represent our management's beliefs and assumptions only as of the date of this presentation. These statements are not guarantees of future performance and undue reliance should not be placed on them. Such forward-looking statements necessarily involve known and unknown risks, which may cause actual performance and results in future periods to differ materially from any projections expressed or implied herein. Avalanche undertakes no obligation to update forward-looking statements. Although forward-looking statements are our best prediction at the time they are made, there can be no assurance that they will prove to be accurate, as actual results and future events could differ materially. The reader is cautioned not to place undue reliance on forward-looking statements.

The overarching aim of `Avalanche` is to provide a unifying platform for the creation, transfer, and trade of digital assets.

By construction, `Avalanche` possesses the following properties:

Scalable `Avalanche` is designed to be massively scalable, robust, and efficient. The core consensus engine is able to support a global network of potentially hundreds of millions of internet-connected, low and high-powered devices that operate seamlessly, with low latencies and very high transactions per second.

Secure `Avalanche` is designed to be robust and achieve high security. Classical consensus protocols are designed to withstand up to f attackers, and fail completely when faced with an attacker of size $f + 1$ or larger, and Nakamoto consensus provides no security when 51% of the miners are Byzantine. In contrast, `Avalanche` provides a very strong guarantee of safety when the attacker is below a certain threshold, which can be parametrized by the system designer, and it provides graceful degradation when the attacker exceeds this threshold. It can uphold safety (but not liveness) guarantees even when the attacker exceeds 51%. It is the first permissionless system to provide such strong security guarantees.

Decentralized `Avalanche` is designed to provide unprecedented decentralization. This implies a commitment to multiple client implementations and no centralized control of any kind. The ecosystem is designed to avoid divisions between classes of users with different interests. Crucially, there is no distinction between miners, developers, and users.

Governable and Democratic `$AVAX` is a highly inclusive platform, which enables anyone to connect to its network and participate in validation and first-hand in governance. Any token holder can have a vote in selecting key financial parameters and in choosing how the system evolves.

Interoperable and Flexible `Avalanche` is designed to be a universal and flexible infrastructure for a multitude of blockchains/assets, where the base `$AVAX` is used for security and as a unit of account for exchange. The system is intended to support, in a value-neutral fashion, many blockchains to be built on top. The platform is designed from the ground up to make it easy to port existing blockchains onto it, to import balances, to support multiple scripting languages and virtual machines, and to meaningfully support multiple deployment scenarios.

Outline The rest of this paper is broken down into four major sections. Section 2 outlines the details of the engine that powers the platform. Section 3 discusses the architectural model behind the platform, including subnetworks, virtual machines, bootstrapping, membership, and staking. Section 4 explains the governance model that enables dynamic changes to key economic parameters. Finally, in Section 5 explores various peripheral topics of interest, including potential optimizations, post-quantum cryptography, and realistic adversaries.

Naming Convention The name of the platform is Avalanche, and is typically referred to as "the Avalanche platform", and is interchangeable/synonymous with "the Avalanche network", or – simply – Avalanche. Codebases will be released using three numeric identifiers, labeled "v.[0-9].[0-9].[0-100]", where the first number identifies major releases, the second number identifies minor releases, and the third number identifies patches. The first public release, codenamed Avalanche Borealis, is v. 1.0.0. The native token of the platform is called "$AVAX". The family of consensus protocols used by the Avalanche platform is referred to as the Snow* family. There are three concrete instantiations, called Avalanche, Snowman, and Frosty.

2 The Engine

Discussion of the Avalanche platform begins with the core component which powers the platform: the consensus engine.

Background Distributed payments and – more generally – computation, require agreement between a set of machines. Therefore, consensus protocols, which enable a group of nodes to achieve agreement, lie at the heart of blockchains, as well as almost every deployed large-scale industrial distributed system. The topic has received extensive scrutiny for almost five decades, and that effort, to date, has yielded just two families of protocols: classical consensus protocols, which rely on all-to-all communication, and Nakamoto consensus, which relies on proof-of-work mining coupled with the longest-chain-rule. While classical consensus protocols can have low latency and high throughput, they do not scale to large numbers of participants, nor are they robust in the presence of membership changes, which has relegated them mostly to permissioned, mostly static deployments. Nakamoto consensus protocols [5, 7, 4], on the other hand, are robust, but suffer from high confirmation latencies, low throughput, and require constant energy expenditure for their security.

The Snow* family of protocols, introduced by Avalanche, combine the best properties of classical consensus protocols with the best of Nakamoto consensus. Based on a lightweight network sampling mechanism, they achieve low latency and high throughput without needing to agree on the precise membership of the system. They scale well from thousands to millions of participants with direct participation in the consensus protocol. Further, the protocols do not make use of PoW mining, and therefore avoid its exorbitant energy expenditure and subsequent leak of value in the ecosystem, yielding lightweight, green, and quiescent protocols.

Mechanism and Properties The Snow* protocols operate by repeated sampling of the network. Each node polls a small, constant-sized, randomly chosen set of neighbors, and switches its proposal if a supermajority supports a different value. Samples are repeated until convergence is reached, which happens rapidly in normal operations.

We elucidate the mechanism of operation via a concrete example. First, a transaction is created by a user and sent to a validating node, which is a node participating in the consensus procedure. It is then propagated out to other nodes in the network via gossiping. What happens if that user also issues a conflicting

transaction, that is, a doublespend? To choose amongst the conflicting transactions and prevent the double-spend, every node randomly selects a small subset of nodes and queries which of the conflicting transactions the queried nodes think is the valid one. If the querying node receives a supermajority response in favor of one transaction, then the node changes its own response to that transaction. Every node in the network repeats this procedure until the entire network comes to consensus on one of the conflicting transactions.

Surprisingly, while the core mechanism of operation is quite simple, these protocols lead to highly desirable system dynamics that make them suitable for large-scale deployment.

- *Permissionless, Open to Churn, and Robust.* The latest slew of blockchain projects employ classical consensus protocols and therefore require full membership knowledge. Knowing the entire set of participants is sufficiently simple in closed, permissioned systems, but becomes increasingly hard in open, decentralized networks. This limitation imposes high security risks to existing incumbents employing such protocols. In contrast, Snow* protocols maintain high safety guarantees even when there are well-quantified discrepancies between the network views of any two nodes. Validators of Snow* protocols enjoy the ability to validate without continuous full membership knowledge. They are, therefore, robust and highly suitable for public blockchains.
- *Scalable and Decentralized* A core feature of the Snow family is its ability to scale without incurring fundamental tradeoffs. Snow protocols can scale to tens of thousands or millions of nodes, without delegation to subsets of validators. These protocols enjoy the best-in-class system decentralization, allowing every node to fully validate. First-hand continuous participation has deep implications for the security of the system. In almost every proof-of-stake protocol that attempts to scale to a large participant set, the typical mode of operation is to enable scaling by delegating validation to a subcommittee. Naturally, this implies that the security of the system is now precisely as high as the corruption cost of the subcommittee. Subcommittees are furthermore subject to cartel formation.
  In Snow-type protocols, such delegation is not necessary, allowing every node operator to have a first-hand say in the system, at all times. Another design, typically referred to as state sharding, attempts to provide scalability by parallelizing transaction serialization to independent networks of validators. Unfortunately, the security of the system in such a design becomes only as high as the easiest corruptible independent shard. Therefore, neither subcommittee election nor sharding are suitable scaling strategies for crypto platforms.
- *Adaptive.* Unlike other voting-based systems, Snow* protocols achieve higher performance when the adversary is small, and yet highly resilient under large attacks.
- *Asynchronously Safe.* Snow* protocols, unlike longest-chain protocols, do not require synchronicity to operate safely, and therefore prevent double-spends even in the face of network partitions. In Bitcoin, for example, if synchronicity assumption is violated, it is possible to operate to independent forks of the Bitcoin network for prolonged periods of time, which would invalidate any transactions once the forks heal.
- *Low Latency.* Most blockchains today are unable to support business applications, such as trading or daily retail payments. It is simply unworkable to wait minutes, or even hours, for confirmation of transactions. Therefore, one of the most important, and yet highly overlooked, properties of consensus protocols is the time to finality. Snow* protocols reach finality typically in ≤ 1 second, which is significantly lower than both longest-chain protocols and sharded blockchains, both of which typically span finality to a matter of minutes.

– *High Throughput.* Snow* protocols, which can build a linear chain or a DAG, reach thousands of transactions per second (5000+ tps), while retaining full decentralization. New blockchain solutions that claim high TPS typically trade off decentralization and security and opt for more centralized and insecure consensus mechanisms. Some projects report numbers from highly controlled settings, thus misreporting true performance results. The reported numbers for $AVAX are taken directly from a real, fully implemented Avalanche network running on 2000 nodes on AWS, geo-distributed across the globe on low-end machines. Higher performance results (10,000+) can be achieved through assuming higher bandwidth provisioning for each node and dedicated hardware for signature verification. Finally, we note that the aforementioned metrics are at the base-layer. Layer-2 scaling solutions immediately augment these results considerably.

Comparative Charts of Consensus Table 1 describes the differences between the three known families of consensus protocols through a set of 8 critical axes.

	Nakamoto	Classical	Snow*
Robust (Suitable for Open Settings)	+	-	+
Highly Decentralized (Allows Many Validators)	+	-	+
Low Latency and Quick Finality (Fast Transaction Confirmation)	-	+	+
High Throughput (Allows Many Clients)	-	+	+
Lightweight (Low System Requirements)	-	+	+
Quiescent (Not Active When No Decisions Performed)	-	+	+
Safety Parameterizable (Beyond 51% Adversarial Presence)	-	-	+
Highly Scalable	-	-	+

Table 1. Comparative chart between the three known families of consensus protocols. Avalanche, Snowman, and Frosty all belong to the Snow* family.

3 Platform Overview

In this section, we provide an architectural overview of the platform and discuss various implementation details. The Avalanche platform cleanly separates three concerns: chains (and assets built on top), execution environments, and deployment.

3.1 Architecture

Subnetworks A subnetwork, or subnet, is a dynamic set of validators working together to achieve consensus on the state of a set of blockchains. Each blockchain is validated by one subnet, and a subnet can validate arbitrarily many blockchains. A validator may be a member of arbitrarily many subnets. A subnet decides who may enter it, and may require that its constituent validators have certain properties. The Avalanche platform supports the creation and operation of arbitrarily many subnets. In order to create a new subnet or to join a subnet, one must pay a fee denominated in $AVAX.

The subnet model offers a number of advantages:

- If a validator doesn't care about the blockchains in a given subnet, it will simply not join that subnet. This reduces network traffic, as well as the computational resources required of validators. This is in contrast to other blockchain projects, in which every validator must validate every transaction, even those they don't care about.
- Since subnets decide who may enter them, one can create private subnets. That is, each blockchain in the subnet is validated only by a set of trusted validators.
- One can create a subnet where each validator has certain properties. For example, one could create a subnet where each validator is located in a certain jurisdiction, or where each validator is bound by some real-world contract. This may be benificial for compliance reasons.

There is one special subnet called the Default Subnet. It is validated by all validators. (That is, in order to validate any subnet, one must also validate the Default Subnet.) The Default Subnet validates a set of pre-defined blockchains, including the blockchain where `$AVAX` lives and is traded.

Virtual Machines Each blockchain is an instance of a Virtual Machine (VM.) A VM is a blueprint for a blockchain, much like a class is a blueprint for an object in an object-oriented programming language. The interface, state and behavior of a blockchain is defined by the VM that the blockchain runs. The following properties of a blockchain, and other, are defined by a VM:

- The contents of a block
- The state transition that occurs when a block is accepted
- The APIs exposed by the blockchain and their endpoints
- The data that is persisted to disk

We say that a blockchain "uses" or "runs" a given VM. When creating a blockchain, one specifies the VM it runs, as well as the genesis state of the blockchain. A new blockchain can be created using a pre-existing VM, or a developer can code a new one. There can be arbitrarily many blockchains that run the same VM. Each blockchain, even those running the same VM, is logically independent from others and maintains its own state.

3.2 Bootstrapping

The first step in participating in `Avalanche` is bootstrapping. The process occurs in three stages: connection to seed anchors, network and state discovery, and becoming a validator.

Seed Anchors Any networked system of peers that operates without a permissioned (i.e. hard-coded) set of identities requires some mechanism for *peer discovery*. In peer-to-peer file sharing networks, a set of trackers are used. In crypto networks, a typical mechanism is the use of DNS seed nodes (which we refer

to as seed anchors), which comprise a set of well-defined seed-IP addresses from which other members of the network can be discovered. The role of DNS seed nodes is to provide useful information about the set of active participants in the system. The same mechanism is employed in Bitcoin Core [1], wherein the `src/chainparams.cpp` file of the source code holds a list of hard-coded seed nodes. The difference between BTC and Avalanche is that BTC requires just one correct DNS seed node, while Avalanche requires a simple majority of the anchors to be correct. As an example, a new user may choose to bootstrap the network view through a set of well established and reputable exchanges, any one of which individually are *not* trusted. We note, however, that the set of bootstrap nodes does not need to be hard-coded or static, and can be provided by the user, though for ease of use, clients may provide a default setting that includes economically important actors, such as exchanges, with which clients wish to share a world view. There is no barrier to become a seed anchor, therefore a set of seed anchors can not dictate whether a node may or may not enter the network, since nodes can discover the latest network of Avalanche peers by attaching to any set of seed anchors.

Network and State Discovery Once connected to the seed anchors, a node queries for the latest set of state transitions. We call this set of state transitions the *accepted frontier*. For a chain, the accepted frontier is the last accepted block. For a DAG, the accepted frontier is the set of vertices that are accepted, yet have no accepted children. After collecting the accepted frontiers from the seed anchors, the state transitions that are accepted by a majority of the seed anchors is defined to be accepted. The correct state is then extracted by synchronizing with the sampled nodes. As long as there is a majority of correct nodes in the seed anchor set, then the accepted state transitions must have been marked as accepted by at least one correct node.

This state discovery process is also used for network discovery. The membership set of the network is defined on the validator chain. Therefore, synchronizing with the validator chain allows the node to discover the current set of validators. The validator chain will be discussed further in the next section.

3.3 Sybil Control and Membership

Consensus protocols provide their security guarantees under the assumption that up to a threshold number of members in the system could be adversarial. A Sybil attack, wherein a node cheaply floods the network with malicious identities, can trivially invalidate these guarantees. Fundamentally, such an attack can only be deterred by trading off presence with proof of a hard-to-forge resource [3]. Past systems have explored the use of Sybil deterrence mechanisms that span proof-of-work (PoW), proof-of-stake (PoS), proof-of-elapsed-time (POET), proof-of-space-and-time (PoST), and proof-of-authority (PoA).

At their core, all of these mechanisms serve an identical function: they require that each participant have some "skin in the game" in the form of some economic commitment, which in turn provides an economic barrier against misbehavior by that participant. All of them involve a form of stake, whether it is in the form of mining rigs and hash power (PoW), disk space (PoST), trusted hardware (POET), or an approved identity (PoA). This stake forms the basis of an economic cost that participants must bear to acquire a voice. For instance, in Bitcoin, the ability to contribute valid blocks is directly proportional to the hash-power of the proposing participant. Unfortunately, there has also been substantial confusion between consensus protocols

versus Sybil control mechanisms. We note that the choice of consensus protocols is, for the most part, orthogonal to the choice of the Sybil control mechanism. This is not to say that Sybil control mechanisms are drop-in-replacements for each other, since a particular choice might have implications about the underlying guarantees of the consensus protocol. However, the Snow* family can be coupled with many of these known mechanisms, without significant modification.

Ultimately, for security and to ensure that the incentives of participants are aligned for the benefit of the network, `$AVAX` choose PoS to the core Sybil control mechanism. Some forms of stake are inherently centralized: mining rig manufacturing (PoW), for instance, is inherently centralized in the hands of a few people with the appropriate know-how and access to the dozens of patents required for competitive VLSI manufacturing. Furthermore, PoW mining leaks value due to the large yearly miner subsidies. Similarly, disk space is most abundantly owned by large datacenter operators.Further, all sybil control mechanisms that accrue ongoing costs, e.g. electricity costs for hashing, leak value out of the ecosystem, not to mention destroy the environment. This, in turn, reduces the feasibility envelope for the token, wherein an adverse price move over a small time frame can render the system inoperable. Proof-of-work inherently selects for miners who have the connections to procure cheap electricity, which has little to do with the miners' ability to serialize transactions or their contributions to the overall ecosystem. Among these options, we choose proof-of-stake, because it is green, accessible, and open to all. We note, however, that while the `$AVAX` uses PoS, the `Avalanche` network enables subnets to be launched with PoW and PoS.

Staking is a natural mechanism for participation in an open network because it enables a direct economic argument: the probability of success of an attack is directly proportional to a well-defined monetary cost function. In other words, the nodes that stake are economically motivated to not engage in behavior that might hurt the value of their stake. Additionally, this stake does not incur any additional upkeep costs (other then the opportunity cost of investing in another asset), and has the property that, unlike mining equipment, is fully consumed if used in a catastrophic attack. For PoW operations, mining equipment can be simply reused or – if the owner decides to – entirely sold back to the market.

A node wishing to enter the network can freely do so by first putting up a stake that is immobilized during the duration of participation in the network. The user determines the amount duration of the stake. Once accepted, a stake cannot be reverted. The main goal is to ensure that nodes substantially share the same mostly stable view of the network. We anticipate setting the minimum staking time on the order of a week.

Unlike other systems that also propose a PoS mechanism, `$AVAX` does not make usage of slashing, and therefore all stake is returned when the staking period expires. This prevents unwanted scenarios such as a client software or hardware failure leading to a loss of coins. This dovetails with our design philosophy of building predictable technology: the staked tokens are not at risk, even in the presence of software or hardware flaws.

In `Avalanche`, a node that wants to participate issues a special *stake transaction* to the validator chain. Staking transactions name an amount to stake, the staking key of the participant that is staking, the duration, and the time that validation will start. Once the transaction is accepted, the funds will be locked until the end of the staking period. The minimal allowed amount is decided and enforced by the system. The stake amount placed by a participant has implications for both the amount of influence the participant has in the

consensus process, as well as the reward, as discussed later. The specified staking duration, must be between δ_{min} and δ_{max}, the minimum and maximum timeframes for which any stake can be locked. As with the staking amount, the staking period also has implications for the reward in the system. Loss or theft of the staking key cannot lead to asset loss, as the staking key is used only in the consensus process, not for asset transfer.

3.4 Smart Contracts in $AVAX

At launch Avalanche supports standard Solidity-based smart contracts through the Ethereum virtual machine (EVM). We envision that the platform will support a richer and more powerful set of smart contract tools, including:

- Smart contracts with off-chain execution and on-chain verification.
- Smart contracts with parallel execution. Any smart contracts that do not operate on the same state in any subnet in Avalanche will be able to execute in parallel.
- An improved Solidity, called Solidity++. This new language will support versioning, safe mathematics and fixed point arithmetic, an improved type system, compilation to LLVM, and just-in-time execution.

If a developer requires EVM support but wants to deploy smart contracts in a private subnet, they can spin-up a new subnet directly. This is how Avalanche enables functionality-specific sharding through the subnets. Furthermore, if a developer requires interactions with the currently deployed Ethereum smart contracts, they can interact with the Athereum subnet, which is a spoon of Ethereum. Finally, if a developer requires a different execution environment from the Ethereum virtual machine, they may choose to deploy their smart contract through a subnet that implements a different execution environment, such as DAML or WASM. Subnets can support additional features beyond VM behavior. For example, subnets can enforce performance requirements for bigger validator nodes that hold smart contracts for longer periods of time, or validators that hold contract state privately.

4 Governance and The $AVAX Token

4.1 The $AVAX Native Token

Monetary Policy The native token, $AVAX, is capped-supply, where the cap is set at $720,000,000$ tokens, with $360,000,000$ tokens available on mainnet launch. However, unlike other capped-supply tokens which bake the rate of minting perpetually, $AVAX is designed to react to changing economic conditions. In particular, the objective of $AVAX's monetary policy is to balance the incentives of users to stake the token versus using it to interact with the variety of services available on the platform. Participants in the platform collectively act as a decentralized reserve bank. The levers available on Avalanche are staking rewards, fees, and airdrops, all of which are influenced by governable parameters. Staking rewards are set by on-chain governance, and are ruled by a function designed to never surpass the capped supply. Staking can be induced by increasing fees or increasing staking rewards. On the other hand, we can induce increased engagement with the Avalanche platform services by lowering fees, and decreasing the staking reward.

Uses

Payments True decentralized peer-to-peer payments are largely an unrealized dream for the industry due to the current lack of performance from incumbents. `$AVAX` is as powerful and easy to use as payments using Visa, allowing thousands of transactions globally every second, in a fully trustless, decentralized manner. Furthermore, for merchants worldwide, `$AVAX` provides a direct value proposition over Visa, namely lower fees.

Staking: Securing the System On the `Avalanche` platform, sybil control is achieved via staking. In order to validate, a participant must lock up coins, or stake. Validators, sometimes referred to as stakers, are compensated for their validation services based on staking amount and staking duration, amongst other properties. The chosen compensation function should minimize variance, ensuring that large stakers do not disproportionately receive more compensation. Participants are also not subject to any "luck" factors, as in PoW mining. Such a reward scheme also discourages the formation of mining or staking pools enabling truly decentralized, trustless participation in the network.

Atomic swaps Besides providing the core security of the system, the `$AVAX` token serves as the universal unit of exchange. From there, the `Avalanche` platform will be able to support trustless atomic swaps natively on the platform enabling native, truly decentralized exchanges of any type of asset directly on `Avalanche`.

4.2 Governance

Governance is critical to the development and adoption of any platform because – as with all other types of systems – `Avalanche` will also face natural evolution and updates. `$AVAX` provides on-chain governance for critical parameters of the network where participants are able to vote on changes to the network and settle network upgrade decisions democratically. This includes factors such as the minimum staking amount, minting rate, as well as other economic parameters. This enables the platform to effectively perform dynamic parameter optimization through a crowd oracle. However, unlike some other governance platforms out there, `Avalanche` does not allow unlimited changes to arbitrary aspects of the system. Instead, only a pre-determined number of parameters can be modified via governance, rendering the system more predictable and increasing safety. Further, all governable parameters are subject to limits within specific time bounds, introducing hysteresis, and ensuring that the system remains predictable over short time ranges.

A workable process for finding globally acceptable values for system parameters is critical for decentralized systems without custodians. `Avalanche` can use its consensus mechanism to build a system that allows anyone to propose special transactions that are, in essence, system-wide polls. Any participating node may issue such proposals.

Nominal reward rate is an important parameter that affects any currency, whether digital or fiat. Unfortunately, cryptocurrencies that fix this parameter might face various issues, including deflation or inflation. To that end, the nominal reward rate is subject to governance, within pre-established boundaries. This will allow token holders to choose on whether `$AVAX` is eventually capped, uncapped, or even deflationary.

Transaction fees, denoted by the set $\mathcal{F}$, are also subject to governance. $\mathcal{F}$ is effectively a tuple which describes the fees associated with the various instructions and transactions. Finally, staking times and amounts are also governable. The list of these parameters is defined in Figure 1.

- Δ : Staking amount, denominated in $AVAX. This value defines the minimal stake required to be placed as bond before participating in the system.
- δ_{min} : The minimal amount of time required for a node to stake into the system.
- δ_{max} : The maximal amount of time a node can stake.
- $\rho : (\pi\Delta, \tau\delta_{min}) \rightarrow \mathbb{R}$: Reward rate function, also referred to as minting rate, determines the reward a participant can claim as a function of their staking amount given some number of π publicly disclosed nodes under its ownership, over a period of τ consecutive δ_{min} timeframes, such that $\tau\delta_{min} \leq \delta_{max}$.
- $\mathcal{F}$: the fee structure, which is a set of governable fees parameters that specify costs to various transactions.

Fig. 1. Key non-consensus parameters used in Avalanche. All notation is redefined upon first use.

In line with the principle of predictability in a financial system, governance in $AVAX has hysteresis, meaning that changes to parameters are highly dependent on their recent changes. There are two limits associated with each governable parameter: time and range. Once a parameter is changed using a governance transaction, it becomes very difficult to change it again immediately and by a large amount. These difficulty and value constraints relax as more time passes since the last change. Overall, this keeps the system from changing drastically over a short period of time, allowing users to safely predict system parameters in the short term, while having strong control and flexibility for the long term.

5 Discussion

5.1 Optimizations

Pruning Many blockchain platforms, especially those implementing Nakamoto consensus such as Bitcoin, suffer from perpetual state growth. This is because – by protocol – they have to store the entire history of transactions. However, in order for a blockchain to grow sustainably, it must be able to prune old history. This is especially important for blockchains that support high performance, such as Avalanche.

Pruning is simple in the Snow* family. Unlike in Bitcoin (and similar protocols), where pruning is not possible per the algorithmic requirements, in $AVAX nodes do not need to maintain parts of the DAG that are deep and highly committed. These nodes do not need to prove any past history to new bootstrapping nodes, and therefore simply have to store active state, i.e. the current balances, as well as uncommitted transactions.

Client Types Avalanche can support three different types of clients: archival, full, and light. Archival nodes store the entire history of the $AVAX subnet, the staking subnet, and the smart contract subnet, all the

way to genesis, meaning that these nodes serve as bootstrapping nodes for new incoming nodes. Additionally these nodes may store the full history of other subnets for which they choose to be validators. Archival nodes are typically machines with high storage capabilities that are paid by other nodes when downloading old state. Full nodes, on the other hand, participate in validation, but instead of storing all history, they simply store the active state (e.g. current UTXO set). Finally, for those that simply need to interact securely with the network using the most minimal amount of resources, `Avalanche` supports light clients which can prove that some transaction has been committed without needing to download or synchronize history. Light clients engage in the repeated sampling phase of the protocol to ensure safe commitment and network wide consensus. Therefore, light clients in `Avalanche` provide the same security guarantees as full nodes.

Sharding Sharding is the process of partitioning various system resources in order to increase performance and reduce load. There are various types of sharding mechanisms. In network sharding, the set of participants is divided into separate subnetworks as to reduce algorithmic load; in state sharding, participants agree on storing and maintaining only specific subparts of the entire global state; lastly, in transaction sharding, participants agree to separate the processing of incoming transactions.

In `Avalanche Borealis`, the first form of sharding exists through the subnetworks functionality. For example, one may launch a gold subnet and another real-estate subnet. These two subnets can exist entirely in parallel. The subnets interact only when a user wishes to buy real-estate contracts using their gold holdings, at which point `Avalanche` will enable an atomic swap between the two subnets.

5.2 Concerns

Post Quantum Cryptography Post-quantum cryptography has recently gained widespread attention due to the advances in the development of quantum computers and algorithms. The concern with quantum computers is that they can break some of the currently deployed cryptographic protocols, specifically digital signatures. The `Avalanche` network model enables any number of VMs, so it supports a quantum-resistant virtual machine with a suitable digital signature mechanism. We anticipate several types of digital signature schemes to be deployed, including quantum resistant RLWE-based signatures. The consensus mechanism does not assume any kind of heavy crypto for its core operation. Given this design, it is straightforward to extend the system with a new virtual machine that provides quantum secure cryptographic primitives.

Realistic Adversaries The Avalanche paper [6] provides very strong guarantees in the presence of a powerful and hostile adversary, known as a round-adaptive adversary in the full point-to-point model. In other terms, the adversary has full access to the state of every single correct node *at all times*, knows the random choices of all correct nodes, as well as can update its own state at any time, before and after the correct node has the chance to update its own state. Effectively, this adversary is all powerful, except for the ability to directly update the state of a correct node or modify the communication between correct nodes. Nonetheless, in reality, such an adversary is purely theoretical since practical implementations of the strongest possible adversary are limited at statistical approximations of the network state. Therefore, in practice, we expect worst-case-scenario attacks to be difficult to deploy.

Inclusion and Equality A common problem in permissionless currencies is that of the "rich getting richer". This is a valid concern, since a PoS system that is improperly implemented may in fact allow wealth generation to be disproportionately attributed to the already large holders of stake in the system. A simple example is that of leader-based consensus protocols, wherein a subcommittee or a designated leader collects all the rewards during its operation, and where the probability of being chosen to collect rewards is proportional to the stake, accruing strong reward compounding effects. Further, in systems such as Bitcoin, there is a "big get bigger" phenomenon where the big miners enjoy a premium over smaller ones in terms of fewer orphans and less lost work. In contrast, Avalanche employs an egalitarian distribution of minting: every single participant in the staking protocol is rewarded equitably and proportionally based on stake. By enabling very large numbers of people to participate first-hand in staking, Avalanche can accommodate millions of people to participate equally in staking. The minimum amount required to participate in the protocol will be up for governance, but it will be initialized to a low value to encourage wide participation. This also implies that delegation is not required to participate with a small allocation.

6 Conclusion

In this paper, we discussed the architecture of the Avalanche platform. Compared to other platforms today, which either run classical-style consensus protocols and therefore are inherently non-scalable, or make usage of Nakamoto-style consensus that is inefficient and imposes high operating costs, the Avalanche is lightweight, fast, scalable, secure, and efficient. The native token, which serves for securing the network and paying for various infrastructural costs is simple and backwards compatible. $AVAX has capacity beyond other proposals to achieve higher levels of decentralization, resist attacks, and scale to millions of nodes without any quorum or committee election, and hence without imposing any limits to participation.

Besides the consensus engine, Avalanche innovates up the stack, and introduces simple but important ideas in transaction management, governance, and a slew of other components not available in other platforms. Each participant in the protocol will have a voice in influencing how the protocol evolves *at all times*, made possible by a powerful governance mechanism. Avalanche supports high customizability, allowing nearly instant plug-and-play with existing blockchains.

References

1. Bitcoin: bitcoin/bitcoin (Oct 2018), https://github.com/bitcoin/bitcoin
2. Buttolph, S., Moin, A., Sekniqi, K., Sirer, E.G.: Avalanche token paper - token dynamics (2019), https://files.avalabs.org/papers/token.pdf
3. Douceur, J.R.: The sybil attack. In: International Workshop on Peer-to-Peer Systems. pp. 251–260. Springer (2002)
4. Eyal, I., Gencer, A.E., Sirer, E.G., van Renesse, R.: Bitcoin-ng: A scalable blockchain protocol. In: 13th USENIX Symposium on Networked Systems Design and Implementation, NSDI 2016, Santa Clara, CA, USA, March 16-18, 2016. pp. 45–59 (2016), https://www.usenix.org/conference/nsdi16/technical-sessions/presentation/eyal
5. Nakamoto, S.: Bitcoin: A peer-to-peer electronic cash system (2008)
6. Rocket, T.: Snowflake to Avalanche: A novel metastable consensus protocol family for cryptocurrencies. IPFS (2018), https://ipfs.io/ipfs/QmUy4jh5mGNZvLkjies1RWM4YuvJh5o2FYopNPVYwrRVGV

7. Wood, G.: Ethereum: A secure decentralised generalised transaction ledger (2014)