SKALE

WHITEPAPER

# The SKALE Network

An Ethereum Interoperable Elastic Blockchain Network

**This document is for informational purposes only and does not constitute an offer or solicitation to sell shares or securities in the N.O.D.E. Foundation, SKALE Labs, Inc. or any related or associated company. Any such offer or solicitation would only be made by a confidential offering memorandum and in accordance with applicable securities and other laws. Changes and updates will be made over time. Please visit skale.network for the most recent version.**

v3.03 - JUL 2020

# Contents

# Abstract

**This document and the associated technologies are under development and subject to change.**

Decentralized networks allow for a new wave of business models and organizational structures to come to light. Their potential for societal and business impact have been well documented as have their short-comings with regards to performance, usability, and cost-effectiveness. The SKALE Network has been designed with the goal of resolving the technical scalability, user experience, and cost issues affiliated with decentralized networks such as Ethereum. In addition, SKALE is designed to bring application specific architecture to developers resulting in enhanced configurability and modularity.

SKALE proposes a decentralized, configurable network of on-demand blockchains that support high-throughput, low-cost, and low-latency transactions enabled with storage capabilities and advanced analytics. Additionally, SKALE proposes messaging protocols that enable participants to communicate between these disparate systems.

This system aims to provide Ethereum-as-a-Service to developers by providing a gasless subscription-based decentralized network for the provisioning and deployment of high-throughput, EVM-compatible, storage-enabled, provably secure byzantine fault tolerant blockchains.

The primary use for these blockchains are Ethereum-compatible elastic sidechains. Each proof-of-stake sidechain is highly configurable, comprised of nodes which stake SKL tokens on the Ethereum mainnet, and leverages an asynchronous byzantine fault tolerant protocol for its consensus mechanism.

# SKALE

The SKALE Network is a high-throughput, low-latency, configurable byzantine fault tolerant, elastic blockchain network built interoperably with Ethereum. The initial and primary use case for this network will be in form of elastic sidechains for the Ethereum Blockchain. In this context it can be described as an 'Elastic Sidechain Network'.

Sidechains in the network are operated by a group of virtualized subnodes selected from a subset of nodes in the network and are run on all or a subset (multitenancy) of each node's computation and storage resources. Each sidechain is highly configurable, with consumers being able to choose their chain's size, consensus protocol, virtual machine, parent blockchain, and additional security measures (e.g. virtualized subnode shuffling frequency).

The SKALE token is a work and usage token. To have the right to work in the network, nodes must run the SKALE daemon and stake a predetermined amount of SKALE tokens on the Ethereum mainnet via a series of smart contracts known as the SKALE Manager. Once a node has been admitted to the network, 24 peer nodes will be randomly[1] selected to audit its uptime and latency - these metrics will be submitted regularly to the SKALE Manager and will affect a node's rewards for participating in the network.

When creating an Elastic Sidechain, consumers will specify their desired chain configuration and submit payment for the duration that they would like to rent network resources to run the chain. If the network has enough bandwidth, nodes meeting computation and storage requirements specified in the chain's configuration will be randomly assigned to participate as its virtualized subnodes.

EVM-compatibility within Elastic Sidechains allows consumers to deploy existing Ethereum-based smart contracts directly to them while an increased gas limit lifts the computation and storage limitations of the Ethereum mainnet EVM. This allows for much more use cases for smart contracts that were previously impossible to run in a cost-effective or performant manner. As an example, each Elastic Sidechain deploys with a FileStorage smart contract that allows for the ability to store large files (up to 100MB) on nodes in the network - something which would necessitate the filling of ~10,000 blocks to do on the Ethereum mainnet.

Each Elastic Sidechain also supports BLS signatures in its consensus model - allowing for interchain messaging. Virtualized subnodes on each chain are able to validate that a transaction was signed and committed by virtualized subnodes on another chain through the use of that chain's group signature, which is made available to all other chains on the Ethereum mainnet. Such an extension of Elastic Sidechains supports the microservice model where each chain is able to perform a specific operation and feed their outputs as inputs to other Elastic Sidechains.

As each node in the network continues to participate in their assigned Elastic Sidechains, they are awarded bounties based upon their performance (as measured by their peer nodes) at the end of each network epoch[2]. When an Elastic Sidechain has reached the end of its lifetime, the resources (computation, storage) of its virtualized subnodes will be freed so that they may participate in newly created Elastic Sidechains.

[1] Prior to MainNet launch, the SKALE Network will be using random number generation based on SKALE BLS implementation for the source of randomness in the network.
[2] A set period of time within the network during which nodes are evaluated on their latency and uptime by peer nodes.

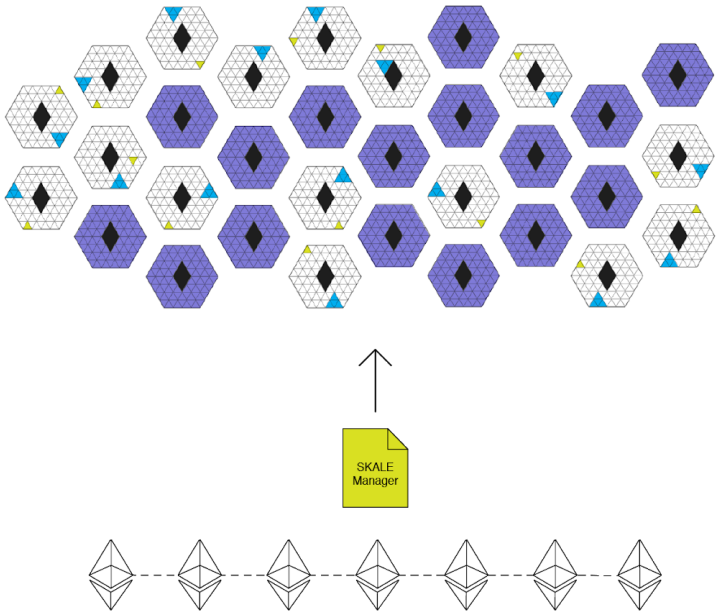
While the SKALE Network is initially only supporting the Ethereum Blockchain, the network will grow to support other Security Layer blockchains over time and serve as an execution and interoperability layer between disparate decentralized technologies.

# SKALE Network

The SKALE Network is comprised of permissionless SKALE Nodes and the SKALE Manager (located on Ethereum).

## SKALE Manager

The SKALE Manager exists on the Ethereum mainnet and serves as the entrypoint to all other smart contracts in the SKALE ecosystem. This contract manages the orchestration of all entities within the network, inclusive of Elastic Sidechain creation / destruction, Node creation / destruction, withdrawals, and bounties.




The SKALE Manager **exists on the Ethereum mainnet and serves as the entrypoint** to all other smart contracts in the SKALE ecosystem.

The SKALE Manager will **leverage the randomness of the Ethereum mainnet** to select a group of nodes with enough space on them to support the desired subnode size and then appoint them to an Elastic Sidechain.

Diagram 1: SKALE Manager

### Node Creation

To be added as a node to the system, a prospective node must run the SKALE daemon which will evaluate the prospective node to ensure that it is upholding network hardware requirements[3]. If the prospective node passes this verification step, the daemon will permit it to submit a request to join the network to the SKALE Manager. This request will contain both the required network deposit as well as node metadata collected by the daemon (e.g. IP address, port, public key,..). After the request has been committed to Ethereum, the

[3] These hardware requirements are subject to change so as to lower barriers to entry by parties interested in running nodes.

prospective node will either be added to the system as 'full node' or a 'fractional node'[4]. Full nodes will have all of their resources utilized for a single Elastic Sidechain while fractional nodes will participate in multiple Elastic Sidechains (multitenancy).

After a node is created, it will have a large group[5] of peer nodes in the network randomly assigned to it; peers regularly audit node downtime and latency at predetermined periods (e.g. five minutes) and submit these batched metrics to the SKALE Manager once for every network epoch where they are used to determine the node's bounty reward.

## Node Destruction

When exiting the network, nodes must first declare their exit and wait a finalization period[6]. After this finalization period (e.g. two days), the node will be inactive and able to withdraw their initial stake from the network.

In the case that a user is unable to wait the finalization period and exits their node immediately from the network, it will be classified as a non-conforming (dead) node by SLA virtualized subnodes, and the bounty for the node will not be paid. It will be then scheduled to be cycled out of the chain.

## Elastic Sidechain Creation

When creating an Elastic Sidechain, consumers select their chain's configuration and submit payment to the SKALE Manager for the duration of time that they wish to rent network resources required to maintain their Elastic Sidechain. To allow users to meet their business / budgetary requirements, they are provided with the option of selecting Elastic Sidechains starting with the minimum of 16 virtualized subnodes whereby each virtualized subnode is either using 1/128 (small), 1/16 (medium), or 1/1 (large) of each node's resources. As the network continues to evolve, it will eventually allow for users to specify the number of virtualized subnodes, number of signers, and size of the virtualized subnodes which will comprise their Elastic Sidechains.

Presently, all resources in the network are of equal value and the cost for consuming these resources is based upon the size of the chain as well as the lifetime of the chain. As the network matures, the cost of network resources will be calculated dynamically to account for current network conditions / system load.

After a creation request has been received by the SKALE Manager, a new Elastic Sidechain will be created and its respective endpoint returned to the creator. If there are not ample[7] resources available in the network to support creation of the desired Elastic Sidechain, the transaction will be canceled and the user will be notified.

---

[4] The ratio of full nodes to fractional nodes will be determined by market demand - the actual algorithm will be specified in the future.

[5] The number of peer nodes is targeted to be 24, but this is subject to change.

[6] Period during which a node's peers are decommissioned and new nodes are appointed to the Elastic Sidechains of the exiting node.

[7] Ample resources refers to enough free resources in the network such that network participants are not able to deduce which nodes will be assigned to an Elastic Sidechain. This means that SKALE ensures that it always has non-trivial amount (>30%) of its resources in the network unoccupied.

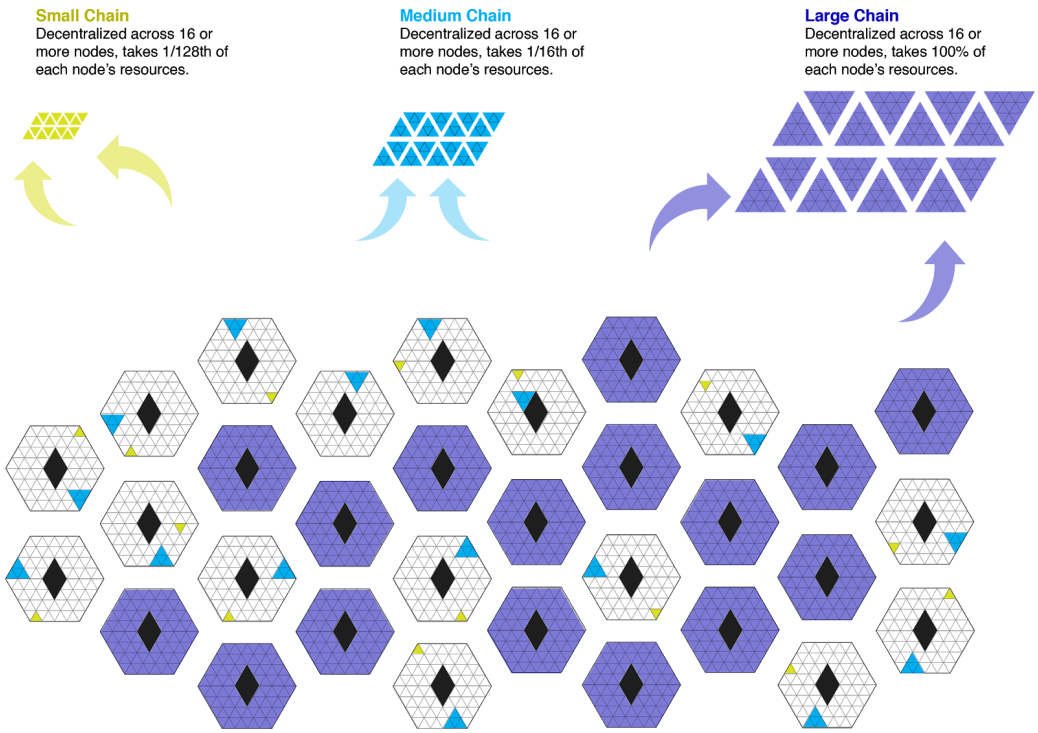



Diagram 2. Elastic Sidechain Creation

## Virtualized Subnode Shuffling

When creating an Elastic Sidechain, developers are provided with the option of enabling virtualized subnode shuffling as an added security measure. Shuffling is encouraged to mitigate any collusion attempts by virtualized subnodes within each Elastic Sidechain and is facilitated through the SKALE Manager in a similar fashion to the node exiting process.

> *Note: To remove the possibility that consumers are able to deduce what nodes are assigned to their Elastic Sidechains during creation or shuffling, the SKALE Network enforces that >30% of total node resources be always left available to serve as the network virtualized subnode validator pool.*

## Elastic Sidechain Destruction

Destruction of an Elastic Sidechain occurs when a consumer's rental deposit for network resources has been exhausted or a consumer has flagged their Elastic Sidechain for deletion. Prior to the exhaustion of their rental deposit, the creator will be notified of their chain's pending deletion and given the opportunity to add additional time to the chain's lifetime.

Once an Elastic Sidechain's rental deposit has been exhausted, it will be eligible for destruction via the SKALE Manager. The destruction process will transfer any crypto assets originating from Ethereum to their owners on the mainnet, remove all virtualized subnodes from the Elastic Sidechain, hard reset their storage and memory, and remove the Elastic Sidechain from the SKALE Manager before rewarding the submitter who commissioned the destruction of the chain.

*Note: The reward earned by the submitter for destroying a chain is slightly greater than the cost of the transaction and serves as an incentivized garbage-collection mechanism for network resources.*

### Bounty Issuance

At the end of each network epoch, the number of SKALE tokens minted for that period are divided equally amongst all nodes which were participating in the network prior to the epoch beginning. The number of these issued tokens which each node can claim are based upon the average of the metrics submitted by 16 of its 24 peers where the top and bottom four metrics are dropped to mitigate any sort of collusion or malicious intent by peer nodes. Any tokens which are not issued to nodes as a result of poor uptime / latency will be issued to the N.O.D.E. Foundation.

## Elastic Sidechains

The initial implementation of the SKALE Network will offer Elastic Sidechains comprised of virtualized subnodes which engage in block creation and commitment through an asynchronous, leaderless, and provably secure protocol. Such a protocol was designed to exhibit robustness in the case of virtualized subnode downtime where each latent / down virtualized subnode is regarded as a slow link.

So long as >⅔ of the total virtualized subnode validator set are online, they will continue creating and committing new blocks to the chain. This protocol is a multi-phase process illustrated by the following diagram and detailed in the following sections.

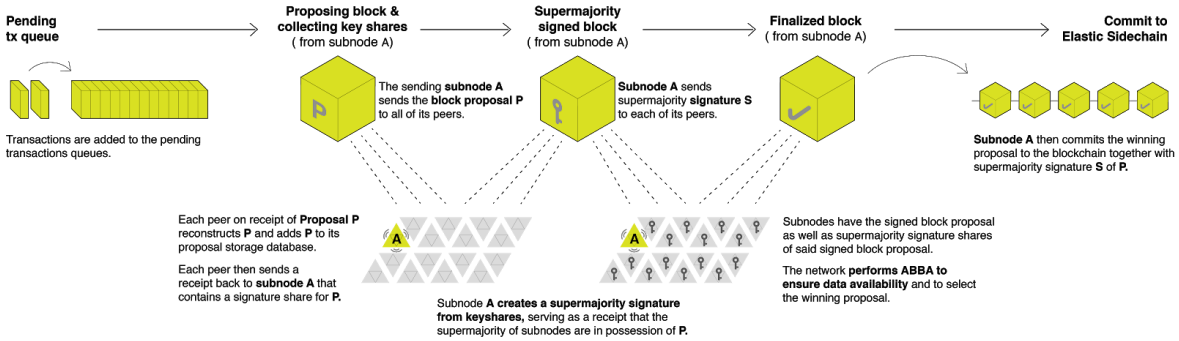



Diagram 3. SKALE Consensus

### Messaging

#### Network Security Assumptions

The protocol assumes that the network is asynchronous with eventual delivery guarantee, meaning that all virtualized subnodes are assumed to be connected by a reliable communications link - links can be arbitrarily slow, but will eventually deliver messages.

This asynchronous model is similar to Bitcoin and Ethereum blockchains and reflects the state of modern Internet, where temporary network splits are normal, but eventually resolve. The eventual delivery guarantee is achieved in practice by the sending virtualized subnode making multiple attempts with exponential backoff to transfer the message to the receiving virtualized subnode, until the transfer is successful.

### Pending Transactions Queue

Each virtualized subnode maintains pending transactions[8] queue. The first virtualized subnode to receive a transaction into that queue will attempt to propagate it to its peers via dedicated outgoing message queues for each. To schedule a message for delivery to a particular peer, it is placed into the corresponding outgoing queue. Each of these outgoing queues is serviced by a separate thread, allowing messages to be delivered in parallel so that failure of a particular peer to accept messages will not affect receipt of messages by other peers.

## Consensus

### Block Proposal

After the previous consensus round has been completed, each virtualized subnode's TIP_ID will increment by 1 and immediately cause them to create a block proposal.

To create a block proposal, a virtualized subnode will:

1. Examine its pending transaction queue.
2. If the total size of transactions in the pending queue is less than or equal to the MAX_BLOCK_SIZE, the virtualized subnode will fill in a block proposal by taking all transactions from the queue.
3. In the case that the total size of transactions in the pending queue exceeds MAX_BLOCK_SIZE, the virtualized subnode will fill in a block proposal of MAX_BLOCK_SIZE by taking pending transactions from queue in order of oldest to newest received.
4. The virtualized subnode will assemble block proposals with transactions which are ordered by SHA-256 Merkle root from smallest value to largest value.
5. In the case that the pending queue is empty, the virtualized subnode will wait for BEACON_TIME, and then, if the queue is still empty, make an empty block proposal containing no transactions.

   *Note: Virtualized subnodes do not remove transactions from the pending queue at the time of proposal. The reason for this is that at the proposal time there is no guarantee that the proposal will be accepted.*

### Data Availability

Once a virtualized subnode creates a block proposal it will communicate it to other virtualized subnodes using the data availability protocol described below. The data availability protocol guarantees that the message is transferred to the supermajority of virtualized subnodes.

The five-step protocol is described below:

1. The sending virtualized subnode A sends both the block proposal and the hashes of the transactions which compose the proposal P to all of its peers.

[8] Each user transaction is assumed to be an Ethereum-compatible transaction, represented as a sequence of bytes.

2. Upon receipt, each peer will reconstruct P from hashes by matching hashes to transactions in its pending queue. For transactions not found in the pending queue, the peer will send a request to the sending virtualized subnode A. The sending virtualized subnode A will then send the bodies of these transactions to the receiving virtualized subnode, allowing for the peer to reconstruct the block proposal and add the proposal to its proposal storage database PD.
3. The peer then sends a receipt to back to A that contains a threshold signature share for P.
4. A will wait until it collects signature shares from a supermajority (>⅔) of virtualized subnodes (including itself). A will then create a supermajority signature S. This signature serves as a receipt that a supermajority of virtualized subnodes are in possession of P.
5. A will then broadcast this supermajority signature S to each of the other virtualized subnodes in the network.

   *Note: Each virtualized subnode is in possession of BLS private key share PKS[I]. Initial generation of key shares is performed using Joint-Feldman Distributed Key Generation (DKG) algorithm which occurs at the creation of the Elastic Sidechain and whenever virtualized subnodes are shuffled.*

In further consensus steps, a data availability receipt is required by all virtualized subnodes voting for proposal P whereby they must include supermajority signature S in their vote; honest virtualized subnodes will ignore all votes that do not include the supermajority signature S. This protocol guarantees data availability, meaning that any proposal P which wins consensus will be available to any honest virtualized subnodes.

### Pluggable Binary Byzantine Agreement

The consensus described below uses an Asynchronous Binary Byzantine Agreement (ABBA) protocol. We currently use a variant of ABBA derived from Mostefaoui et al. Any other ABBA protocol P can be used, as long as it satisfies the following properties:

- Network model: P assumes asynchronous network messaging model described above.
- Byzantine nodes: P assumes less than one third of Byzantine nodes.
- Initial vote: P assumes that each node makes an initial vote yes(1) or no(0)
- Consensus vote: P terminates with a consensus vote of either yes or no, where if the consensus vote is yes, it is guaranteed that at least one honest node voted yes.

  *Note: An ABBA protocol typically outputs a random number COMMON_COIN as a byproduct of its operation. We use this COMMON_COIN as a random number source.*

### Consensus Round

Immediately after the proposal phase completes, each virtualized subnode A who has received supermajority signature S for their proposal P will vote for Asynchronous Byzantine Binary Agreements (ABBAs) in a consensus round R. The protocol is as follows:

1. For each R, virtualized subnodes will execute N instances of ABBA.
2. Each ABBA[i] corresponds to a vote on block proposal from the virtualized subnode i.
3. Each ABBA[i] completes with a consensus vote of yes or no.
4. Once all ABBA[i] complete, there is a vote vector v[i], which includes yes or no for each proposal.
5. If there is only one yes vote, the corresponding block proposal P is committed to the Elastic Sidechain.
6. If there are multiple yes votes, P is pseudorandomly picked from the yes-voted proposals using pseudorandom number R. The winning proposal index the remainder of division of R by N_WIN, where N_WIN is the total number of yes proposals.

7. The random number R is the sum of all ABBA COMMON_COINs.
8. In the rare case where all votes are no, an empty block is committed to the blockchain. The probability of an all-no vote is very small and decreases as N increases.

### Finalizing Winning Block Proposal

Once consensus completes with winning block proposal P on any virtualized subnode A, the virtualized subnode will execute the following algorithm to finalize the proposal and commit it to the chain:

1. A will check if it has received the winning proposal P.
2. If A has not received the proposal, it will request it from its peer virtualized subnodes for download.
3. A will then sign a signature share S for P and send it to all other virtualized subnodes.
4. A will then wait to receive signature shares from a supermajority of virtualized subnodes, including itself.
5. Once A has received a supermajority of signature shares, it will combine them into a threshold signature.
6. A will then commit the P to the blockchain together with the threshold signature S.

Blocks which are committed to the Elastic Sidechain contain a block header and block body. The block body is a concatenated transactions array of all transactions in the block and the block header is a JSON object which includes the following:

| **Name** | **Data Type** | **Description** |
|---|---|---|
| BLOCK_ID | integer | ID of the current block, starting from 0 and incremented by 1. |
| BLOCK_PROPOSER | integer | ID of the node that proposed the block. |
| PREVIOUS_BLOCK_HASH | string | SHA-256 Merkle root of the previous block. |
| CURRENT_BLOCK_HASH | string | SHA-256 Merkle root of the current block. |
| TRANSACTION_COUNT | integer | Count of transactions in the current block. |
| TRANSACTION_SIZES | integer[] | An array of transaction sizes in the current block. |
| CURRENT_BLOCK_PROPOSER_SIG | string | ECDSA signature of the proposer of the current block. |
| CURRENT_BLOCK_TSIG | integer | BLS supermajority threshold signature of the current block. |

Table 1. SKALE Block Header Format

## SKALE Virtualized Subnodes

Each Elastic Sidechain is comprised of a collective of randomly appointed virtualized subnodes which run the SKALE daemon and run SKALE consensus. Unlike other protocols, virtualized subnodes are not restricted to a

one-to-one mapping between participating nodes in the network. This is made possible through the containerized virtualized subnode architecture deployed on each node in the SKALE Network which allows each node to run multiple Elastic Sidechains simultaneously.

Subnodes within a SKALE Node are referred to as Virtualized Subnodes. Each Virtualized Subnode participates in independent Elastic Sidechains. Below is a diagram of a container running on a SKALE virtualized subnode.

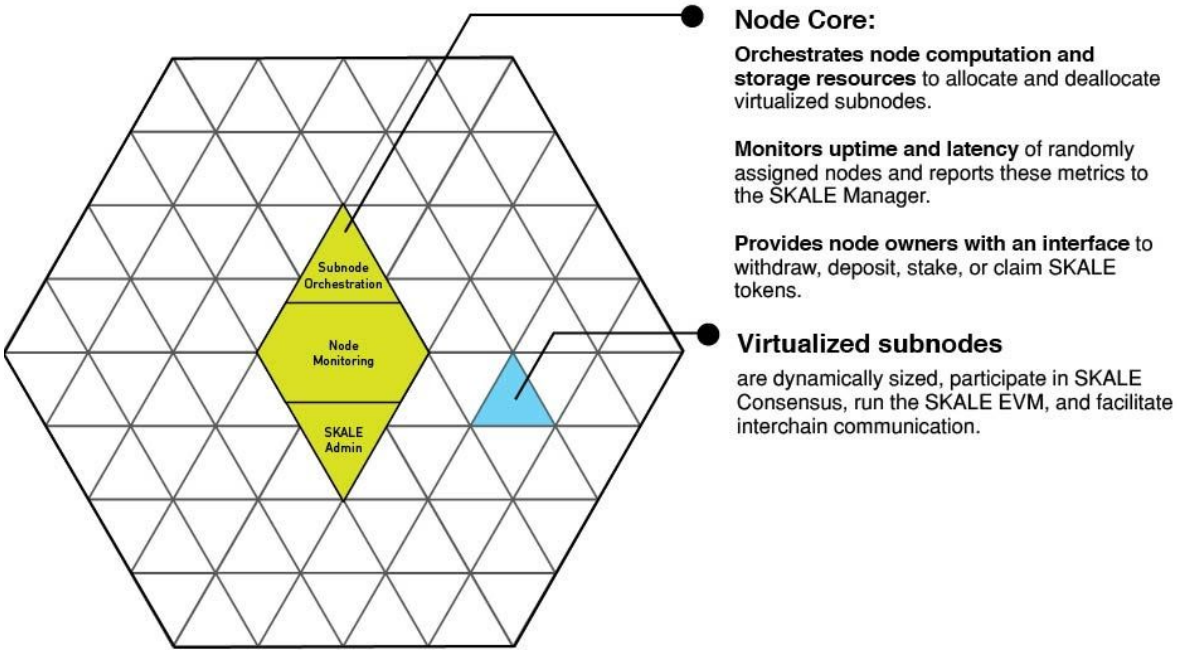



Diagram 4. SKALE Node

This containerized architecture was selected as a means to bring enterprise grade performance and optionality to decentralized application developers on par with centralized systems, which offer elasticity, configurability, and modularity. Containers are divided into five main components that ship with a dockerized Linux OS - allowing for each node to be hosted in a OS-agnostic manner. Each container is encapsulated within one of the following services.

## SKALE Admin Service

The SKALE Admin Service serves as the human-facing interface for virtualized subnodes with the SKALE Manager (located on the Ethereum Mainnet). Functionality shipping with this interface includes the ability for nodes to see which Elastic Sidechains they are participating in as well as the ability to deposit, withdraw, stake, and claim[9] SKALE tokens. Because virtualized subnodes within nodes are appointed randomly to participate in Elastic Sidechains, there is no interface for being able to join / leave Elastic Sidechains within the network.

[9] SKALE tokens available to be claimed are those which are regularly issued at to nodes based upon their average uptime / latency for a network epoch.

## Node Monitoring Service

The NMS (Node Monitoring Service) is run on each SKALE Node and facilitates the performance tracking of each of that node's peer nodes. Performance tracking is measured in both uptime and latency through a regular process which pings each peer node and logs these measurements to a local database. At the end of each Elastic Sidechain epoch, these metrics will be averaged and submitted to the SKALE Manager which will use them to determine the payout to each node.

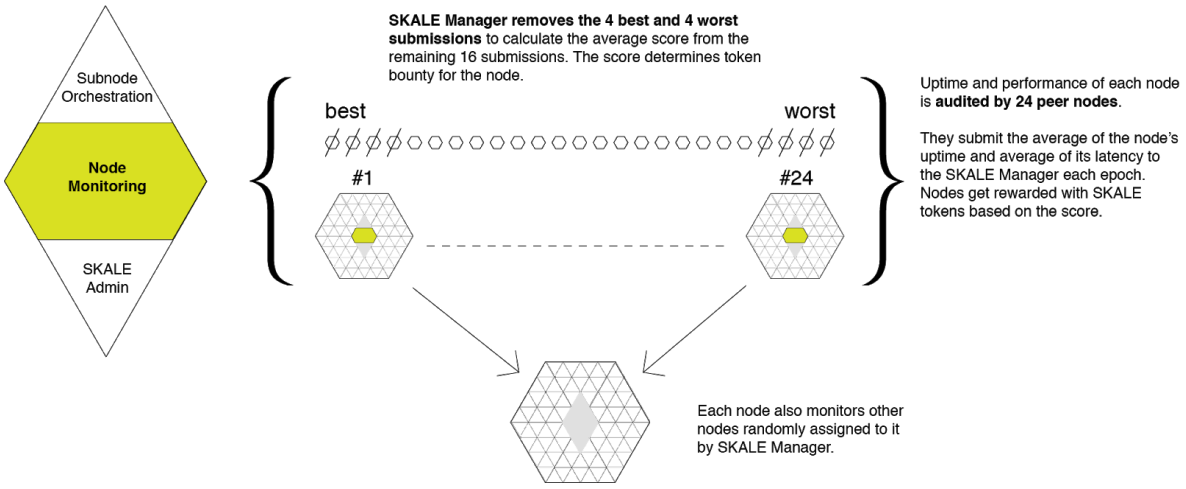



Diagram 5. Node Monitoring Service

## Virtualized Subnode Orchestration Service

The VSOS (Virtualized Subnode Orchestration Service) orchestrates node computation and storage resources to instantiate virtualized subnodes using a dynamically created virtualized subnode image consisting of the SKALE daemon (skaled), the Catchup Agent for syncing an Elastic Sidechain, and the transfer agent for interchain messaging. This service also performs respawning of failed virtualized subnodes as well as deallocation of resources to virtualized subnodes who have been decommissioned.

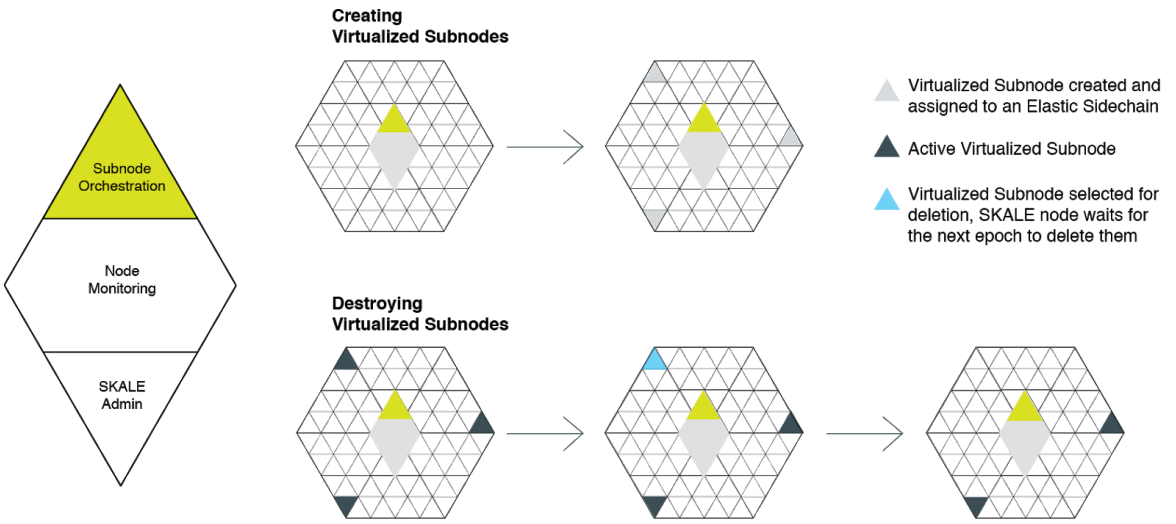



Diagram 6. Virtualized Subnode Orchestration Service

## Attacks and Faults

To account for network downtime, SKALE has integrated a series of contingency strategies for performing fault recovery on both the node and chain level. These range from an automated agent for performing recovery for a downed node to a security incident response team available to all Elastic Sidechain operators in the network.

### Reboots / Crashes

During a reboot, the rebooting node will become temporarily unavailable — for peer nodes, this will look like a temporarily slow network link. After a reboot, messages destined to the node will be delivered — this protocol allows for a reboot to occur without disrupting the operation of consensus.

In the case of a hard crash where a node loses consensus state due to a hardware failure or a software bug that prevents the node from being online, its peers will continue attempting to send messages to it until their outgoing messages queues overflow — causing them to drop older messages. To mitigate the effects of this, messages older than one hour are targeted to be dropped from message queues.

While a node is undergoing a hard crash, it is counted as a Byzantine node for each consensus round — allowing for <⅓ of nodes to be experiencing hard crashes simultaneously. In the case where >⅓ nodes experience a hard crash, consensus will stall, causing the blockchain to possibly lose its liveness.

Such a catastrophic failure will be detected through the absence of new block commits for a set time period. At this point, a failure recovery protocol utilizing the Ethereum main chain for coordination will be executed. Nodes will stop their consensus operation, sync their blockchains, and agree on a time to restart consensus. Finally, after a period of mandatory silence, nodes will start consensus at an agreed point.

### Catchup Agent

A separate Catchup Agent running on each node is responsible for ensuring that node's blockchain and block proposal database are synced with the network. The catchup engine is continuously making random sync

connections to other nodes whereby any node discovering that they have a smaller TIP_ID than their peer will download the missing blocks, verify supermajority threshold signatures on the received blocks, and commit them to its chain.

When the node comes online from a hard crash, it will immediately start this catchup procedure while simultaneously participating in the consensus for new blocks by accepting block proposals and voting according to consensus mechanism but without issuing its own block proposals. The reason for this is that each block proposal requires the hash of the previous block, and a node will only issue its own block proposal for a particular BLOCK_ID once it has finished the catch up procedure.

With such an agent running on each node, nodes having experienced a hard crash will be able to easily rejoin in block proposal after re-syncing their chains.

### Security Incident Response

Security is a foremost requirement for all decentralized systems, but despite progress in cryptography and computer science, most security experts agree that perfect security is unattainable. With this in mind, architects need to concentrate on raising the bar as much as possible for the amount of resources and money required to break the system.

Since the SKALE architecture is based on Elastic Sidechains, a security compromise of SKALE could involve the compromise of a particular Elastic Sidechain. For example, a significant number of nodes could be affected by a computer virus due to a bug in the Linux kernel. In such a case, the default procedure is as follows:

1. Elastic Sidechain owners suspecting of a security compromise will issue a request to the Ethereum SKALE Manager contract to temporarily suspend the chain.
2. The SKALE Manager will mark the Elastic Sidechain as suspended.
3. Uncompromised virtualized subnodes will receive a notification from the SKALE Manager to freeze their operation.
4. Clients of the Elastic Sidechain will be notified of the suspension and have their requests to the Elastic Sidechain rejected.
5. The Elastic Sidechain creator will have an opportunity to consult with the N.O.D.E. Foundation Security Incident Response Team (SIRT) to resolve the issue.

SIRT members are community security experts nominated and elected by SKALE stakeholders who, once elected, will receive a modest compensation from N.O.D.E. Foundation.

Typical incident response will be to identify an uncompromised node and clone its Elastic Sidechain content to a new, uncompromised Elastic Sidechain. Once a new Elastic Sidechain has been established, the consensus operation will be restarted and clients of the Elastic Sidechain will be notified. After the investigation is completed, SIRT will have the power to slash the security deposits of offending nodes.

# SKALE Protocol

## Next Gen PoS-Based Network

To encourage proper behavior amongst network participants, SKALE follows a Proof-of-Stake system whereby each node must stake a predetermined amount of SKALE tokens to be slashed at the citation of any activity not

condoned by the network. These activities generally include those which denote a failure to properly participate[10] in each assigned chain's consensus and maintain uptime and latency standards enforced by network-agreed-upon SLAs.

Network SLAs are enforced through an algorithmic peer review system whereby each node is appointed 24 peers to monitor and log their network participation, uptime, and latency. These metrics will be collected and averaged on the Ethereum mainnet to reward or slash nodes according to their respective performance.

The requirement of staking a nontrivial sum of SKALE tokens to participate in the network also serves as a measure in sybil-resistance to thwart any adversarial attempt (>⅔ of all network resources would be required to launch a successful attack).

## Delegation

Holders of SKALE are provided the option to delegate their tokens to any node in the network that does not already have the maximum number of tokens staked / delegated. These delegated tokens will receive markedly less[11] of a reward each network epoch than the virtualized subnode.

Due to the leaderless nature of SKALE's consensus, virtualized subnode weighting will have no effect on the rewards gained by each node nor the way in which virtualized subnodes propose / commit new blocks to each chain.

Further delegation details will be released in later versions of the whitepaper.

## Consensus

When building consensus algorithms, it is important to account for malicious actors within the network, botnets, Distributed Denial of Service (DDoS) attacks, malicious firewalls, etc… which can all interfere with network messaging. At the same time, it's crucial that any large-scale network be able to support high-throughput of messages and account for downtime of nodes within the network.

For these reasons, SKALE currently uses a variant of Moustefaoi et. al consensus as it provides a number of highly desirable and necessary properties for a truly decentralized, high-throughput network. This protocol allows for a leaderless, asynchronous, and byzantine fault tolerant network.

### Leaderless

In many existing decentralized / distributed consensus protocols, there is a leader elected for each round to propose some data ("a block") for the network to run consensus and reach agreement on. The SKALE consensus protocol instead implements a protocol whereby all virtualized subnodes are able to propose blocks and only those which receive a supermajority of signatures ("a threshold") are eligible to be accepted for potential commitment to the blockchain.

Leaderlessness serves not only to prevent collusion amongst network participants, but also serves to ensure that all virtualized subnodes participating have a fair chance to propose a block within a chain.

[10] Nodes in the SKALE network will be slashed upon continued demonstration of Byzantine behavior or refusal to sign blocks.

[11] These rewards are to be determined by a network delegation standard.

### Asynchronous

In an asynchronous timing model, there are no bounds or expectations placed upon how long it will take for a message to be delivered within the network. Virtualized subnodes sending messages in the network do so with no immediate expectation of a response and implement an exponential backoff procedure whereby they attempt to redeliver messages that have not been responded to with longer intervals between them. This model accurately captures the current state of how the Internet functions, where nodes in the network fail and messages are dropped all the time.

### Byzantine Fault Tolerant

Byzantine Fault Tolerance (BFT) is the standard for security in distributed systems; systems which are BFT guarantee that nodes in a network can always agree on the same consensus in the present of <⅓ malicious nodes. Nodes deemed as 'malicious' in the network may be exhibiting a variety of behaviors not limited to lying, collusion, and non-participation.

Amongst various implementations of BFT, those which are asynchronous (ABFT) are amongst the strongest. This is for the fact that these account for the possibility that some messages between honest participants are being delayed or not being delivered to their intended recipients - something not uncommon in an Internet-like setting.

### Threshold Signatures

Our protocol uses threshold signatures for supermajority voting. Upon creation of an Elastic Sidechain, Boneh Lynn Shacham (BLS) private key shares PKS[I] are created using joint-Feldman Distributed Key Generation (DKG) and issued to each virtualized subnode. For each PKS[I], there exists a verifiable public key PK[I] which is stored and made publicly available on the SKALE Manager for signature verification purposes.

BLS threshold signatures are implemented as described in Boldyreva with elliptic curve (altBN256) and group pairing (optimal-Ate) implemented in Ethereum Constantinople release.

## Extensions

With this network architecture and protocol, a number of extensions can easily be added to allow for greater functionality and utility of the network. The first two which have been built include enhanced Filestorage within each node and a mechanism for relaying and executing messages between Elastic Sidechains.

### Storage

To expand potential use-cases, SKALE has modified the existing EVM to allow for much larger file storing capabilities. The changes enabling this included the increase of block sizes (allowing for more data to be included in each block) as well as direct access to each node's file system from a fileStorage precompiled smart contract.

Consumers in the network are now able to split files into 1MB "chunks" and submit them to the fileStorage smart contract to be stored on each node's filesystem in a contiguous manner. Files in the network can also be deleted in a rent-style fashion to ensure that the network can reallocate resources due to state bloat from additional storage capabilities.

## Interchain Communication

Availability of Elastic Sidechains' group signatures makes it possible for independent Elastic Sidechains to verify that a block has been signed and committed on another Elastic Sidechain, allowing for the execution of smart contracts as well as the transfer of crypto-assets across Elastic Sidechains. This mechanism is facilitated through a series of smart contracts located on the Ethereum mainnet, each Elastic Sidechain, as well as an agent running on each virtualized subnode which is responsible for facilitating these interchain messages.

Elastic Sidechains each have an inbox and outbox. Messages which are being sent to other chains are kept in the outbox until they are picked up by a randomly-appointed agent who will then send the message to the appropriate recipient chain's inbox as well as any additional metadata[12] which that chain requires to validate that the transaction was included in the sending chain's blockchain. Once this has been proven on the recipient blockchain, the transaction will be forwarded to the destination address / smart contract through an on-chain messaging proxy.

In the case of value transfer from a parent blockchain (ex: Ethereum mainnet), a *DepositBox* is used as a money caching mechanism and two-way peg whereby vouchers are issued against this pooled value on each Elastic Sidechain and exchanged freely amongst participants in the same manner as transactions. When value is exchanged between Elastic Sidechains, the value is first destroyed on the sending chain before creating it on the receiving chain to eliminate the possibility of double-spend attacks. This process is also adhered to for redemption transactions sent to the Ethereum mainnet which free locked capital in the deposit box.

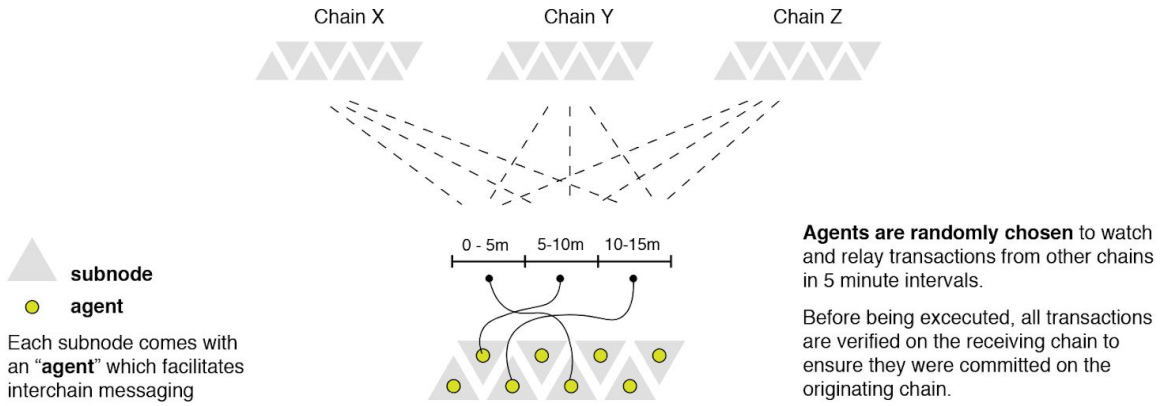



Diagram 7. Interchain Communication

[12] Agents within the network are required to send the committed block in which the transaction was included to the receiving chain so that it is able to verify that the transaction was included in the block and that the block has a valid BLS signature for the current virtualized subnode validator set.

# Governance

Decentralized governance with representation from all core stakeholder groups is critical to the success of the SKALE community and mission. Governance allows all contributors to the N.O.D.E. Foundation to reach important decisions on security, quality, and progress of SKALE network.

SKALE Labs has set up the N.O.D.E. Foundation. SKALE Labs will pass over a collection of IP, money, assets, and power to the Foundation. The Foundation will then release control upon network launch to a decentralized governance model for Network Governance. On-chain voting will control all economics parameters of the SKALE Network. On-chain voting will be supported and made more effective by support of elected Network Representatives and a Foundation Council that are empowered with checks and balances. Ultimately the role of the Council and Network Representatives will be to facilitate community and decentralized control via efficient On-Chain voting.

In general, SKALE governance follows a Delegated Stake Model. A stakeholder can either participate in governance directly by voting with its stake or delegate the voting power to other stakeholders.

The default voting model used by SKALE is a simple majority vote of stakes that participate in the vote. A proposal will typically include a 14-day voting period. SKALE tokens committed to a 90 day "Committed and Formalized Stake" are eligible for voting on key issues brought forth by the SKALE Council which will represent equal seats from the N.O.D.E. Foundation, Mining Community, dApps / Application Community, and Investor Community. Ultimately this council will help optimize and filter topics for On-Chain voting allowing for the positive elements of centralized democracy and decentralized autonomous networks.

# SKALE Token

The SKALE token is a hybrid use token which represents the right to work in the network as a validator, stake as a delegator, or access a share of its resources by deploying and renting an Elastic Sidechain for a period of time as a developer.

Users pay SKALE in a subscription-model to rent these resources (computation, storage, bandwidth) for a predetermined amount of time in the form of an Elastic Sidechain.

Validators stake SKALE into the network and then gain the right to run nodes and earn both fees and tokens via inflation. Delegators may delegate their tokens to validators and earn rewards.

## Distribution

1. Number of tokens
    a. The Total supply of SKL tokens at Network launch is 4,140,000,000. The Network has a Max Supply of 7,000,000,000 tokens.
2. Distribution
    a. 34.3% allocated to the Validator Community and Ecosystem
        i. 33% of which is designed to be paid to validators via inflation at rates listed below.
        ii. ~1.3% will go towards ecosystem community growth via grants and awards, and payment for validator required ecosystem essentials for liquidity of the network tokens.

b. ~25-28%% allocated to Network Supporters who purchase tokens prior to Network Launch with the intent of running Validator Nodes, delegating, or using Elastic Sidechains for their dApps. All are locked for periods of 6-36 months following network launch.
c. ~7.7% are dedicated to support Protocol Development for future financing and grant efforts to support operations and contracts that will improve, enhance, and support the Network.
d. 20% allocated to Network Creators and Builders with a 3-4 year vesting period and 12 month lock both of which commence at Network launch date, putting total vest period at 5-6 years based on Q3 2019 Launch date.
    i. ~16% to broader foundational team and ~4% for Employee Token Option Pool to ensure further development on the network.
e. 10% allocated to N.O.D.E. Foundation. 150 Million are minted at Genesis and 550 Million are minted at Month 6 with an unlocking schedule that commences at 24 months based on milestone achievements which include having an active running network and a decentralized validator community running nodes.
f. 2.5% to the public token event

3. Vesting Schedule and Lock-ups
    a. Pre-purchased tokens in prior SAFT rounds are locked from a period of 9 to 36 months based on SAFT agreements. Lock period commences at network launch.
    b. The team will be locked for one year and will have a 3-4 year vest. Lock and vest time period commences at network launch.
    c. The Foundation will vest over a 7 year period.
4. Inflation
    a. Validator Rewards Mechanism: Validator rewards will mint in the first year at 9.3% of the max supply of tokens. The validator rewards rate will ladder down for the first 6 years then halve every 3 years to perpetuity until the max supply of tokens is reached in the network. These numbers are subject to changes leading up to Mainnet launch based on economic analysis and community feedback.

## Token Distribution Chart

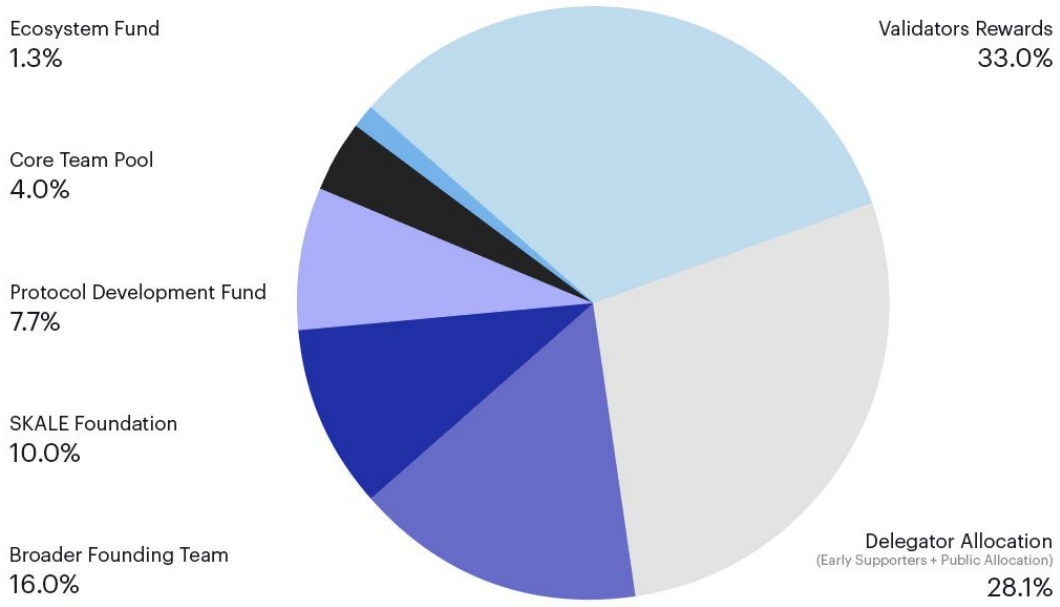



Diagram 8. Token Distribution

## Token Distribution Table

| | |
|---|---|
| Total Supply at Genesis | 4,140,000,000 SKL |
| Maximum Total Supply | 7,000,000,000 SKL |
| Token Type | ERC-777 |

| Segment | Allocation<br>(% of Max Supply) | Lock-Ups and Vesting Schedules |
|---|---|---|
| SKALE Foundation | 10% | 7 year vesting. 150M minted at Network Launch and are unlocked. 550M minted at month 6 based on milestone achievements and are unlocked at 6 month intervals commencing on month 24. |
| Validator Rewards | 33% | Anticipated 9.3% annual network issuance rate, minted monthly. Inflation rate will step function down each year before halving every 3 years on year 6 to perpetuity |
| Ecosystem Fund | ~1.3% | Fund is expected to go live on a predetermined date by the Foundation. |
| Protocol Development Fund | 7.7% | Fund is expected to go live on a predetermined date by the Foundation. |
| Core Team (Broader Founding Team) | 16% | 4 year vest with 12 month lockup commencing upon network launch. Equates to 6+ year total vest based on project commencement date and projected Q320 Mainnet launch. |
| Core Team (Core Team Pool) | 4% | Majority 4 year vest with 12 month lockup (monthly vest) commencing upon network launch. Less than 2% vesting on a 3 year schedule with 1 year lock. Equates to 3-6+ year total vest based on project commencement date and core team employment start dates and projected Q320 Mainnet launch. |
| Early Supporters | 25.6% - 28.2% | See below |
| • Round 1 | 10% | 3 Year lock commencing at network launch with ⅙ of total unlocking every 6 months. Equates to 5+ year total vest based on project commencement date and projected Q320 Mainnet launch. |

| | | |
|---|---|---|
| • Round 2* | 10.5% - 12.8% | 1 year lockup and 3 month unlock commencing at end of 12 month lock out period. Equates to 30+ month total lock based on summer 2018 financing date and projected Q320 Mainnet Launch. |
| • Round 3 | 5.4% | 9 month lock-up followed by 6 month unlock (50% unlocks at month 9 and 50% unlocks at month 15). Equates to a 24+ month total lock based on summer 2019 financing date and projected Q320 Mainnet launch. |
| Public Launch | 4.2% of Total Supply and 2.5% of Max Supply | 2 month required Proof of Use requirement. Equates to 0 days of lock while the token is liquid as tokens will be in transferless state during the Proof of Use period. Exchange listings and token transfers enable following Proof of Use period. Final quantity to be displayed via Activate Dashboard. |

## Token Unlock Schedule

All lock dates commence at Phase 2 Network Launch

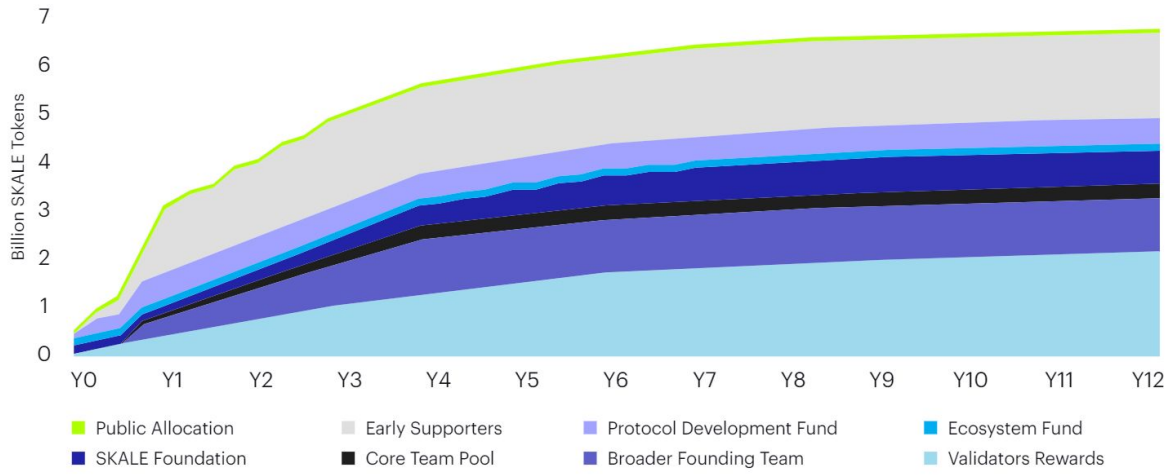



Diagram 9. Token Unlock Schedule

# Summary

The N.O.D.E. Foundation's mission is to bring scale, efficiency, and cost effectiveness to decentralized applications so that the promise of decentralized systems will have a lasting global impact on humanity. We believe the Execution Layer will someday be comprised of hundreds of thousands of Virtualized Subnodes that in conjunction with the Ethereum Ecosystem will run the open, unstoppable internet of the future.

The N.O.D.E. Foundation believes the creation and development of decentralized web infrastructure combined with a passionate, open, and committed community of developers, validators, and advocates will have an exponential impact on the trajectory of blockchains and decentralized organizations.

# Appendix

## SKALE Terminology

| Term | Description |
|---|---|
| ABBA | A randomized agreement protocol which guarantees validity, agreement, and probabilistic termination. |
| ABFT | A type of Byzantine Fault Tolerance which accounts for the possibility that some messages between honest participants are being delayed or not being delivered to their intended recipients. |
| Catchup Agent | The agent shipping with each virtualized subnode responsible for syncing the virtualized subnode with the current 'tip' of the Elastic Sidechain. This enables virtualized subnodes which have faulted to come back online and download / verify the blocks they missed as well was the syncing of new virtualized subnodes who have been shuffled into an existing Elastic Sidechain. |
| DepositBox | The smart contract which facilitates the locking and unlocking of capital on both Elastic Sidechains and the Ethereum mainnet. This mechanism is foundational to the implementation of a two-way peg for value transfer. |
| Messaging Agent | An agent which runs on each virtualized subnode and listens to connected Elastic Sidechains to facilitate interchain messaging. These agents are appointed to listen to incoming messaging at random times to mitigate the possibility of censorship. |
| Parent Blockchain | The blockchain from which value is drawn by way of a one or two-way peg. |
| Peer Node | Nodes in the network which are monitoring the uptime and latency of a specified node. Peer nodes report these metrics to the SKALE Manager at regularly scheduled network epochs. These metrics are used to determine the reward for the specified node. |
| Elastic Sidechain | A fixed set of network virtualized subnodes that accept user transactions, run SKALE consensus and store identical copies of Skale blockchain. A typical implementation will run a network node in a virtualization container such as a Docker container, so a single physical server can provide support to multiple Elastic Sidechains. |
| SKALE Manager | This contract (located on Ethereum) manages the orchestration of all entities within the network, inclusive of Elastic Sidechain creation / destruction, node creation / destruction, withdrawals, and bounties. |
| SKALE Node | Nodes in the SKALE Network which are responsible for the orchestration of resources for the creation and destruction of virtualized subnodes, the monitoring other nodes in the network, and interfacing with the SKALE Manager. |
| Two-Way Peg | Allows the transfer of one crypto-asset from one blockchain to a secondary blockchain and vice-versa. The transfer locks the crypto-asset on one blockchain while unlocking the equivalent amount on a secondary blockchain. The original crypto-asset can be unlocked when the equivalent |

| | |
|---|---|
| | amount of tokens on the secondary blockchain are locked, again. |
| Virtualized Subnode | Participants of Elastic Sidechains who perform SKALE consensus, run the SKALE EVM, and facilitate interchain messaging. |

## SKALE Protocol Parameters

| **Name** | **Description** |
|---|---|
| BEACON_TIME | The time between empty block creation. If no-one is submitting transactions to the blockchain, empty beacon blocks will be created. Beacon blocks are used to detect normal operation of the blockchain. |
| COMMON_COIN | Introduced by Rabin [7], this is a distributed object that delivers the same sequence of random bits b1, b2, . . . , br, . . . to each process. |
| MAX_BLOCK_SIZE | The maximum size of the block body in bytes. Currently, we use 8MB and may consider self-adjusting block size to target a particular average block commit time in the future. |
| N_WIN | The total number of proposals which have been agreed to by virtualized subnodes as valid for commitment and finalization. This number is used in modulo division to decide the index of the winning proposal. |
| TIP_ID | The ID of the latest block (or, “the tip”) in the blockchain. |

# References


1. Achour Mostefaoui , Hamouma Moumen , Michel Raynal, Signature-free asynchronous byzantine consensus with t < n/3 and o(n2) messages, Proceedings of the 2014 ACM symposium on Principles of distributed computing, July 15-18, 2014, Paris, France
2. Alexandra Boldyreva. Threshold Signatures, Multisignatures and Blind Signatures Based on the Gap-Diffie-Hellman-Group Signature Scheme. In Yvo Desmedt, editor, Public Key Cryptography - PKC 2003, volume 2567 of LNCS, pages 31–46. Springer, 2003.
3. P. Feldman. A Practical Scheme for Non-interactive Verifiable Secret Sharing. In FOCS'87, pages 427–437, 1987.
4. Beuchat, J.L., Díaz, J.E.G., Mitsunari, S., Okamoto, E., Rodríguez-Henríquez, F., Teruya, T.: High-speed software implementation of the optimal ate pairing over barreto–naehrig curves. In: Joye, M., Miyaji, A., Otsuka, A. (eds.) Pairing 2010. LNCS, vol. 6487, pp. 21–39. Springer, Heidelberg (2010)
5. Boneh, D., Shacham, H., Lynn, B.: Short Signatures from the Weil Pairing. In: Boyd, C. (ed.) ASIACRYPT 2001. LNCS, vol. 2248, pp. 514–532. Springer, Heidelberg (2001)
6. Christian Cachin , Klaus Kursawe , Victor Shoup, Random Oracles in Constantipole: Practical Asynchronous Byzantine Agreement Using Cryptography (extended abstract), Proceedings of the nineteenth annual ACM symposium on Principles of distributed computing, p.123-132, July 16-19, 2000, Portland, Oregon, USA [doi>10.1145/343477.343531]
7. M. O. Rabin. Randomized Byzantine Generals. In 34th Annual Symposium on Foundations of Computer Science, Palo Alto California, 3-5 November 1993, pages 403–409. IEEE Computer Society, 1983.