Endurio

Yellow Paper

Antony Pham (antony@endur.io), Michael Chu (michael@endur.io)

DRAFT 0.01.01 (2018/09/14)

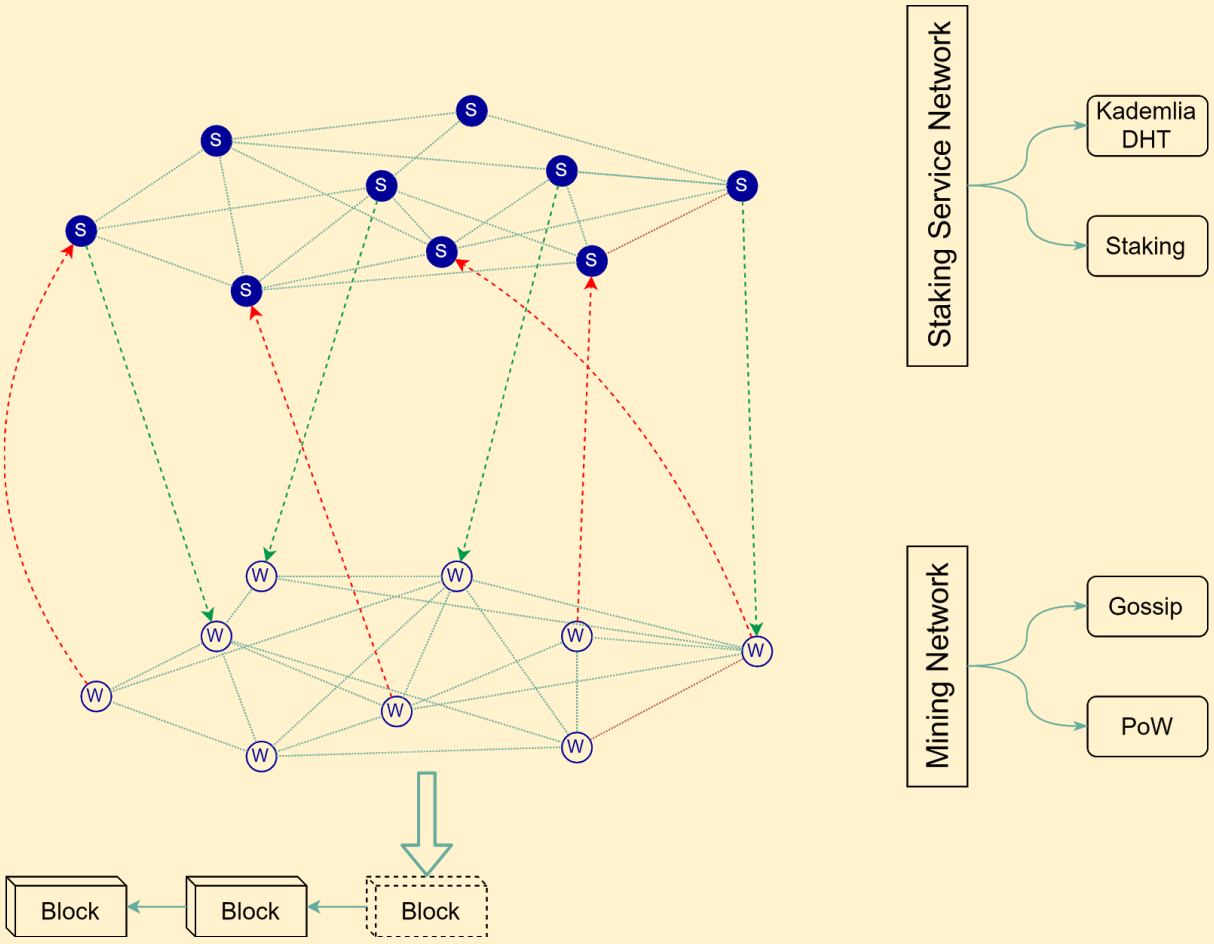
For the most updated version, see:
endur.io

Table of Contents

Staking Service Network

Staking Service Network (SSN) is a second layer network built on top of the Proof of Work layer of a public blockchain, to provide many services that are not possible or inefficient to be implemented in the consensus layer. Those services include (but are not limited to):

- Interactive Services:
    - Input Locking
    - Coin Mixing
- Oracle Service
- Cross-chain Relay Service
- Price Matching Services:
    - Market Order
    - Auction



Proof of Work requires all participants to work on the same job and come to a consensus together, this provides cryptographic security but also limits the performance and scalability of the network. In SSN, only a small number of participants will do the same task together, while the rest of the network do their own tasks. This allows the network to split the workload evenly between each participant, a.k.a network load balancing.

Using the Kademlia P2P network, all nodes in the SSN are addressable by their Public Key, which will be used to verify their signatures for all the service they provide.

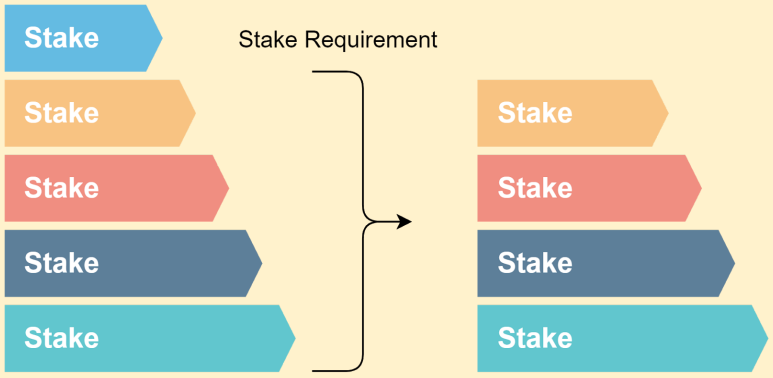
Desirable properties of the SSN:

- Service Request to Service Node mapping is random sampled, so requests cannot choose the serving nodes to cheat.
- Stake-based Frequency: nodes with more staked will have more chance to serve.
- Fault Tolerant: allow up to 49% of the network (stake-based) to be offline or not honest
- Network Load Balancing: requests should be split evenly (stake-based) between participants.
- Nodes with bad behaviour are punishable.

Everyone can join SSN by staking their native crypto-token, to serve the network for a portion of block reward. The staking tokens serve 2 purposes:

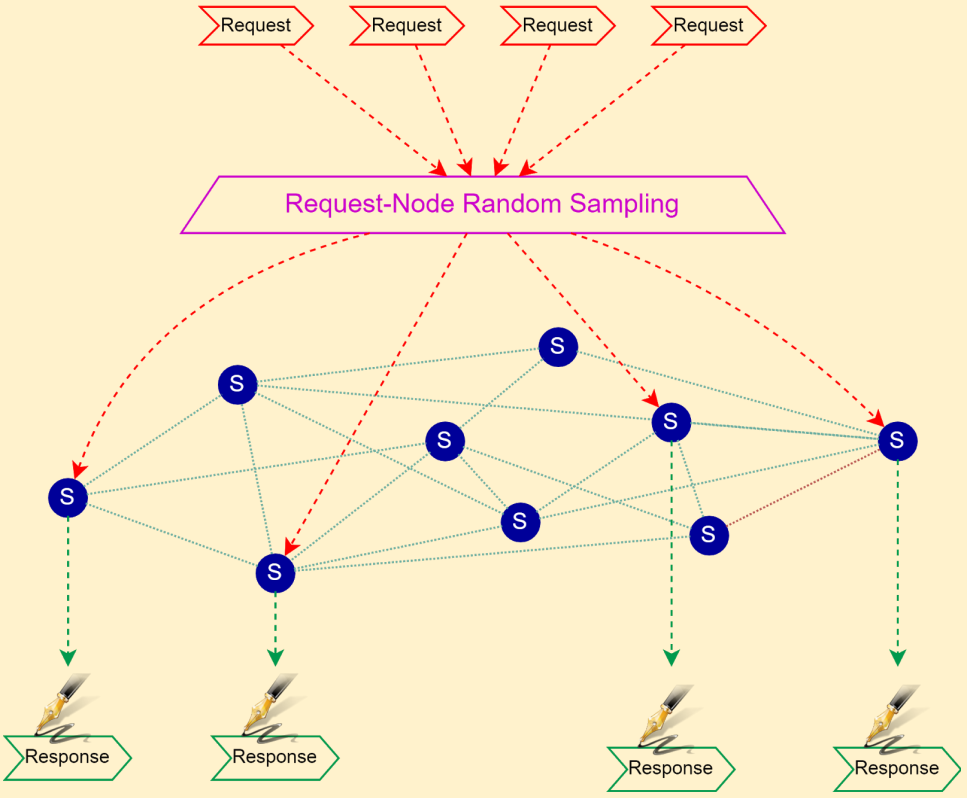
1. Prevent Sybil attack on the SSN.
2. Will be destroyed (or slashed) when a node is detected with bad behaviour.

There is no hard limit of token a node has to stake, but each kind of service has its own limiting mechanism to protect the system. For example, the Input Locking Node can only be selected to lock an input, if its stake is no less than the input UTXO value.



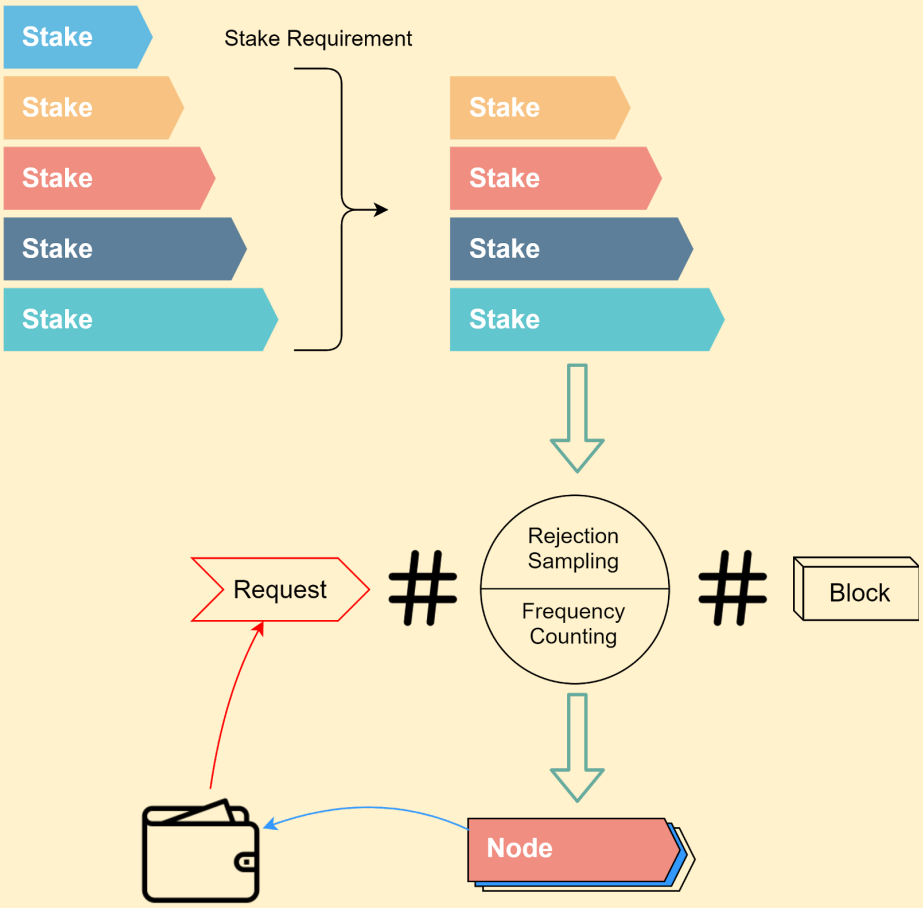
Request-Node Random Sampling

Service requests are served by Service Nodes selected by a random sampling mechanism.



This mapping process has the following properties:

1. Public: each request-node map can be calculated and verified by everybody.
2. Persistence (or Auditability): all data required to calculate the map should be found on-chain, so every service signatures in the block history can always be re-validated to prevent history forging.
3. Secure: it is impossible or extremely costly to *grind* the request data to select favoured nodes.
4. Stake-based: only nodes that meet the Stake Requirement of the Service can be selected and the selection frequency of each node is proportional to its stake.



A typical random sampling process works as follow:

1. All node public-keys are fetched from the blockchain, (since each Staking Node has to stake a portion of token on-chain with a dedicated transaction).
2. Only nodes that meet the Stake Requirement of the Service are kept.
3. The last block hash and the request data are used as entropies.
4. Rejection Sampling or Frequency Counting algorithm is applied on the list to select a serving node.

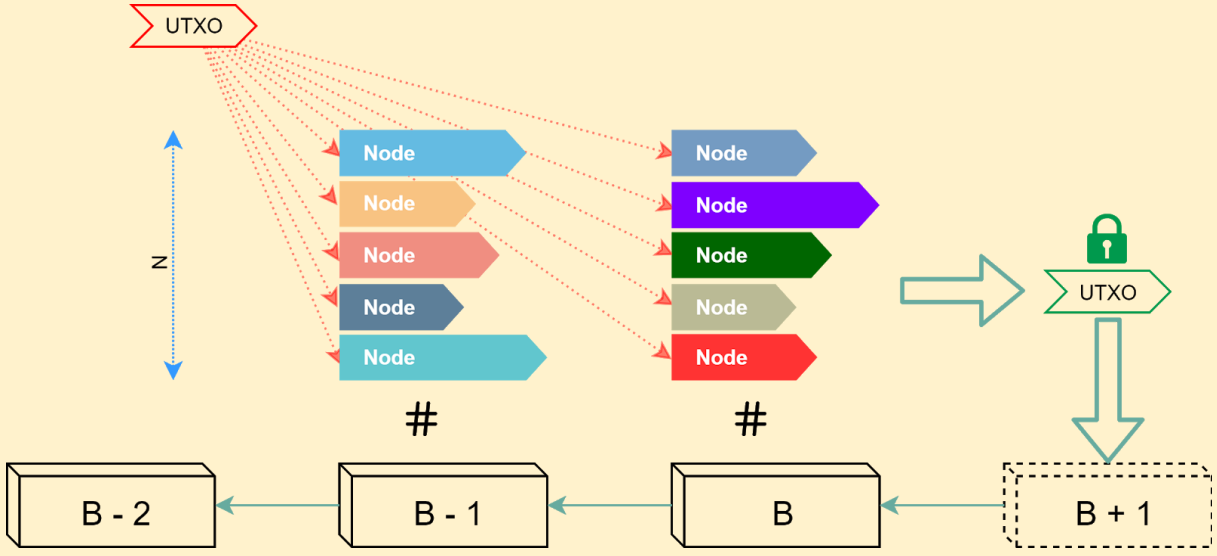
Staking nodes can be offline or maliciously unresponding, so services often select a set of nodes instead of a single one to serve. Larger set provides a higher level of fault tolerant with the cost of higher network footprint.

To increase the availability of the service, one of the most effective strategies is to punish the offline nodes by burning their stake. This can simply be done by burning all staking token and compensate the active nodes by increasing the reward. This compensation in long-term will negate the burning rate of active nodes, while at the same time, punishes the inactive ones, incentivize them to only stake their coins when they are active. Sharding will allow transaction to be so fast and cheap, that nodes will actively stake in and out of the system, as their Service Node go on and off.

Input Locking Service

One of the most desirable services can be implemented in the SSN is the Input Locking Service (ILS). The ILS allows one single node to pledge their stake that a single UTXO can only be included in a particular transaction. That means no double-spending for that UTXO in 3 blocks around the current one.

Using the Request-Node Random Sampling process, a single UTXO is mapped to a single ILS node, so any double-spending detected with that UTXO will contain conflicted signatures by that node. Cheating nodes will have their conflicted signatures included in the next block (by PoW miners) as a Proof of Bad Conduct, and their stake will be used to repay all the possible victims then burnt.

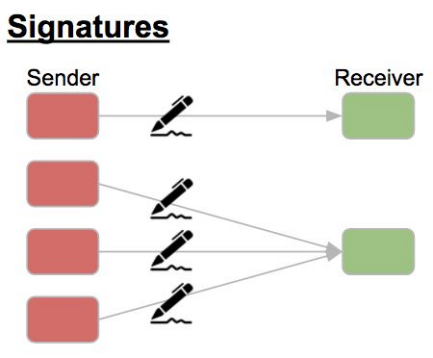


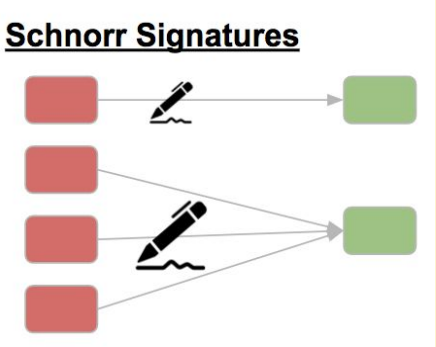
Since service nodes are not always online to serve, and some of them might even be not honest, Input Locking should be Byzantine Fault Tolerant and be working under a certain level of availability. To provide fault tolerance, ILS also maps an UTXO to a set of N nodes instead of one, and the lock is successful only when more than half (N/2+1) of the ILS nodes provide their signatures. Note that, all locking nodes don't have to connect and agree with each other as they verify and lock the transaction individually.

Since current block hash is one of the mapping entropy, a new block mined will re-sampling the whole request-node map. To prevent cross-block double-spending locked by 2 disjoint set of ILS nodes, UTXO has to be signed by both current block and previous block set of nodes. This makes sure the there will be no conflicting lock in 3 blocks time, from 1 block backward to 1 block forward.

Each node with a positive lock will assume the lock is successful, and reject any subsequence lock attempt on the same UTXO. The lock can only be released after 2 blocks time.

A locked transaction with one input UTXO will contain up to 2N+1 signatures, and almost every transactions are locked using this service. Every transaction is effectively a ((N/2+1)*2+1) of (2N+1) multi-sig. With traditional ECDSA, the block size would be drastically increased for the same transaction throughput. But with Schnorr Signature, any amount of signatures in a multisig transaction can be aggregated together to take up only one space in the chain. Thus, the block size is not increased by locked transactions, indeed, it's even smaller due to the smaller size of the Schnorr signature compare to ECDSA signature.





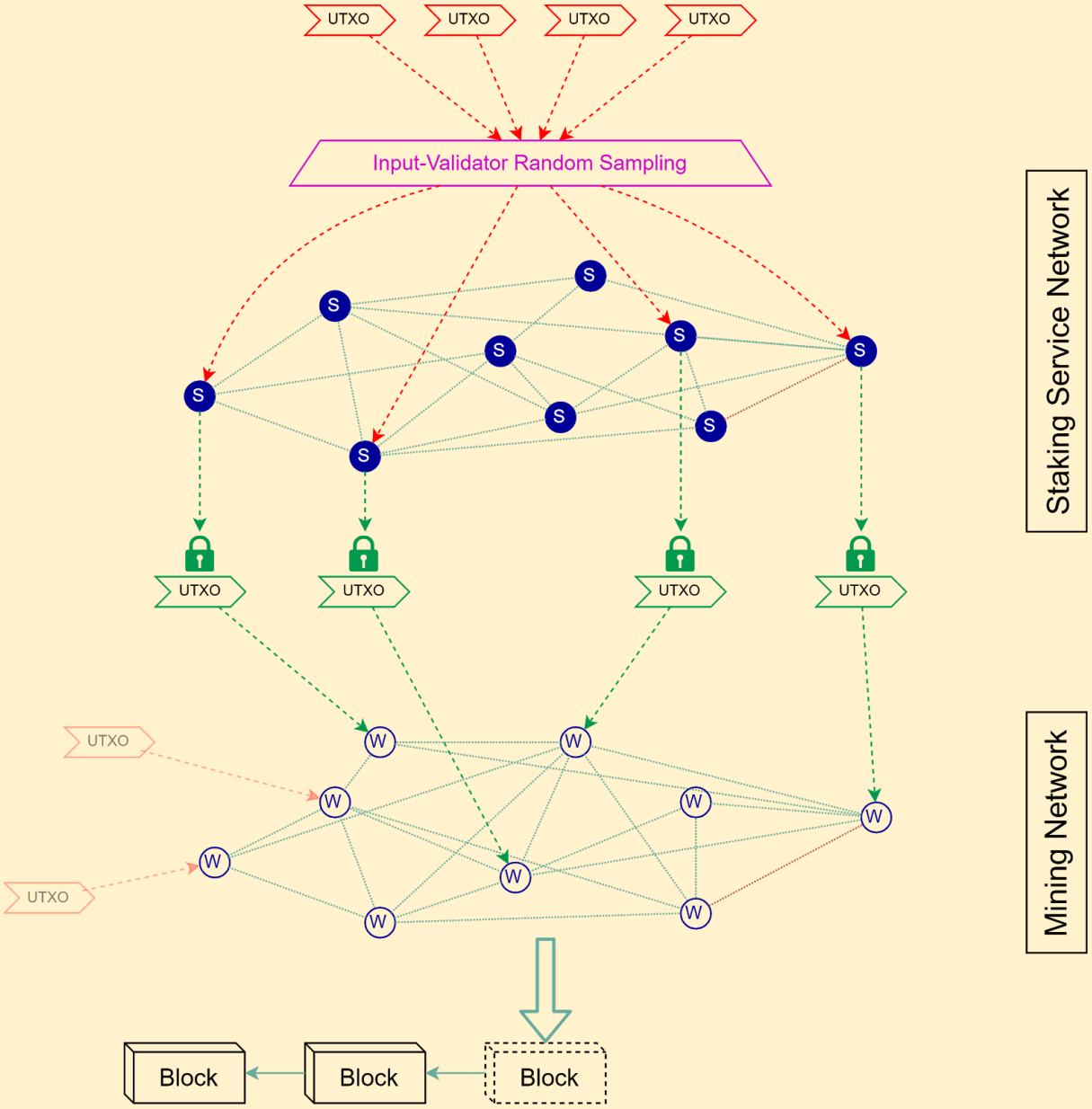
Lock & Block

Proof of Work provides the security and immutability for the blockchain, but the confirmation time is often very long, and even unpredictable. In the sharding context, a pure PoW is also extremely vulnerable to 51% and shard-hopping attacks.

With Input Locking Service, before transactions are propagated to the network, their inputs can be validated and locked by ILS nodes first using their signatures, then packed in a block by PoW miners. When a transaction is Locked for Block #B, then:

1. There will be no conflicting lock in 3 blocks: #B-1, #B and #B+1, *none that cannot be punished*.
2. When properly propagated, it will override any conflicted Open Transaction in #B and #B+1.

In short, when the recipient sees a Locked Transaction propagated to his node, and the latest block is #B-1 or #B, it's almost certain that the transaction will be finalized in the chain.



This secondary mechanism provides the following properties to the network:

- Instant confirmation: A locked transaction will almost certainly be included in a block, and the locking time is only limited by the message traveling speed of the network, which is almost instantly.
- The possibility of double-spend attack is now not of PoW miner, but belong to ILS nodes, which can be punished (or slashed). See Double-spent Attack Penalty.
- The possibility of rewriting history is now extremely hard, even with 51% mining power and 51% of ILS nodes due to the combination of Lock & Block consensus.

Regarding the CAP Theorem, Input Locking service prioritizes Consistency over Availability. That means, sometimes, there will be not enough online ILS nodes to lock a specific input, these cases including:

- The input value is too large, no node in the network has enough stake to be selected.

- Half of the ILS nodes selected for an input are offline. And no another lockable input can be found to replace it.

In those cases, transactions can still be propagated and mined by the good old Proof of Work miners as Open Transactions. Open Transactions should (or need to) wait for at least 2 more blocks for confirmation because they can be overridden by conflicting Locked Transactions in the very next block they were included.

Double-spent Attack Penalty

This part only briefly discusses one of the most common attack vectors to demonstrate how the collateral token of SSN works to punish the node with the bad behavior detected. The full list of attack vectors and SSN protection against them will be detailed in a separated section.

In case of double spent attack detected, where two or more transactions with conflicting UTXO are signed by the same ILS nodes. The following actions will be taken place:

1. Proof of Bad Conduct (all conflicting signatures) will be included on-chain.
2. All conflicting outputs will be paid up to full value, to the best effort. Inputs will be taken from:
   - all available inputs of the transaction remitters, then
   - all staking token of bad nodes.

   The rest of the staking token will be destroyed or kept for later victims.

Because an input can only be locked by ILS nodes with stake no less than the input value, there will always be more stake collateralized to pay for the double-spend attack. This is not true for triple-spend attacks or more if they can be pulled off.

Transaction Overriding

With Locked Transaction introduced, we have a new concept of transaction finality. A transaction with higher finality can override any conflicting transaction with lower finality. The overriding mechanism is design dependent.

The simplest form of overriding design is reverting the block like a hard fork does. Block with the overridden transaction will be invalidated by the consensus, allow new fork with the overriding transaction to take over. This method is simple for non-sharding chain but could be troublesome in a sharding chain, where reverting a shard can cause more depending shards to be reverted.

Another form is on-chain transaction overriding, where:

- low finality transaction cannot be spent before a specific number of confirmations,
- higher finality transaction can override the lower one, by being included in the block right after that and nullify the overridden transaction.

The following list provides different transaction types and their finality:

- Locked Transaction (LTx): highest finality, instant confirmation, pledged by it's ILS nodes. Cannot be double-spent without ILS nodes being punished, the stake of cheating nodes will be used to pay the victims, up to full transaction value.

- Unlockable Transaction (ULTx): transaction with input so large, no ILS node in the network can lock it. Medium finality, same level with Bitcoin transaction, no different.
- Open Transaction (OTx): low finality, requires at least 2 confirmations before its output can be spent. OTx can be overridden by a conflicting LTx before its 3rd confirmation, effectively nullify it.
- Cross-chain and cross-shard Transaction (XTx): very low finality, can be overridden when the counterparty chain/shard get reverted. XTx has to wait for several confirmations before its output can be spent.

Instead of letting higher finality events revert the whole chain, (wasting time and effort of many participants and disrupting the service), transaction overriding allows low-finality transactions to gain more confirmations while waiting for its spending condition. Until the spending condition is reached, OTx and XTx's output cannot be spent, and can always be overridden by other higher finality events.

The main chain is now the heaviest locked chain, which difficulty weight is less favorable for OTx. This prevents the pure 51% PoW attack since the weight counts less for OTx.

Open Transaction (OTx)

Input locking mechanism provides instant confirmation and consistency to the chain, not without a cost. The feature prioritizes Consistency and Decentralization over Availability since there is always chance that half of the selected ILS nodes are offline, or not being honest. Then the input UTXO cannot be locked for at least 2 more blocks. In that case, users have 2 options:

- Try another input, if they're not in extremely bad luck, they will eventually find other lockable input for replacement.
- If no other input can be locked, a transaction can be submitted with a partial lock or none at all. That transaction when included in a block is called "Open Transaction" or OTx.

OTx is low-finality, requires at least 2 confirmations before its output can be spent. OTx can be overridden by a conflicting LTx before its 3rd confirmation, effectively nullify it.

OTx can have a partial lock or none at all. From the protocol view, a partial lock is no different than having no lock. But partial lock can provide more confidence and credibility to transaction recipient, that the transaction sender did actually try to lock it first.

Staking Reward

The block reward is distributed as below:

- 50% for Proof of Work miners,
- 40% for SSN nodes selected for the locks,
- 10% for development subsidy,

All staking coin is bunt at a constant rate, from Start Staking Signal to Stop Staking Signal, whether the node is selected to serve or not. The initial burning rate is 0.01% per block, and

this value is subject to change in the development process. This burning rate is compensated by additional reward for the SSN nodes that serving the block.

Each SSN nodes selected and have their service signature in the block will be rewarded from the staking reward and burning stake, pro-rata. The reward is not distributed on new block mined but on withdrawing request. Each staking node balance is calculated locally on every full node, and only put on-chain when a staking node submits a Staking Reward Withdraw Transaction.

$$r_i = \frac{n_i}{n} \times (R \times 40\% + S \times 0.01\%)$$

r_i = staking reward for node i
n_i = number of node i's signatures in the block
n = total number of all node signatures in the block
R = total block reward (for both PoW and PoS and development subsidy)
S = total stake of running staking service network

As with coinbase transaction for mining rewards, staking reward can only be spent after 146 blocks from the last staking signature. Staking NDR is also locked for 146 blocks after the last served block. This provides a time window for any double-spend victim to submit the conflicting signatures so the cheating SSN nodes can be punished. To unlock the staking NDR, SSN nodes would need to stop serving for a whole day (146 blocks), before the coin can be spent.

Staking Transactions and Signals

The success rate of Input Locking process is directly proportional with the availability of Staking Validators Nodes. To increase this rate along with the availability of the service, one of the most effective strategies is to punish the offline Validator nodes by burning their stake. This is done by burning all staking NDR, and compensate the active nodes by increasing the reward (see Staking Reward).

Staking transactions and signals are also designed to be very compact and efficient, so SSN nodes can actively staking in and out of the system as their nodes go on and off.

Staking Transaction (STx)

To start collateralizing NDR for SSN, a node sends an STx to lock its coin into a special UTXO. This UTXO cannot be spent before 146 blocks from the last block it serves. STx is only submitted once for a while and takes a regular transaction fee.

There is no need for a reverse transaction to unlock the stake, owner of the staked UTXO can spend it after 146 blocks from the last block it serves. The output UTXO will have the value different than the original one, because of the burning and rewarding stake accumulated while the node is serving.

$$s' = s - s \times 0.01\% \times N_{active\ block} + \sum_{b}^{N_{serving\ block}} r_b$$

s' = output stake
s = original input stake
$N_{active\ block}$ = number of blocks that the SSN was active
$N_{serving\ block}$ = number of blocks that the SSN has served
r_b = staking reward in block b for that node

Active blocks are all the blocks between a Start and Stop Staking Signal, inclusively.

Serving blocks are all the blocks that contain at least one serving signature of the SSN node. Those blocks set obviously are a subset of the active blocks set.

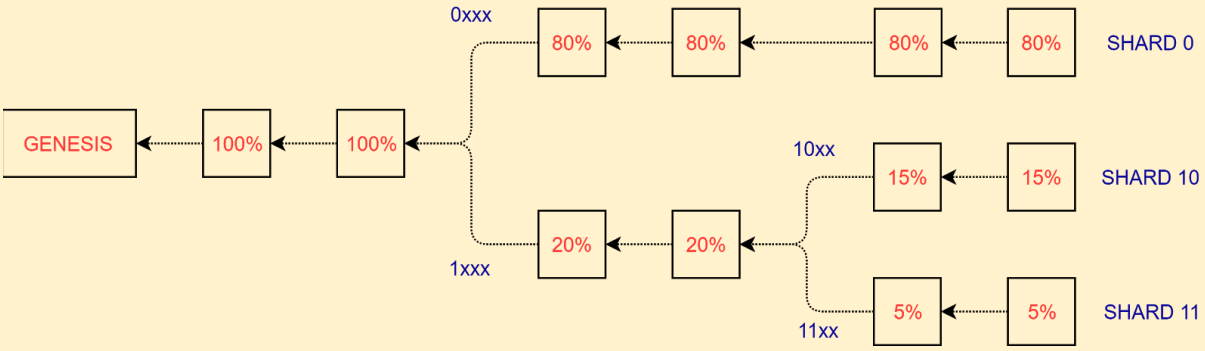
Start and Stop Staking Signals

Start Staking Signal and Stop Staking Signal are two special transactions to signal the network that an SSN node to go online and offline. These two signals only take up to 2 bytes each in the block. Additionally, all signals in the block take only a single aggregated BLS signature for their witnesses.

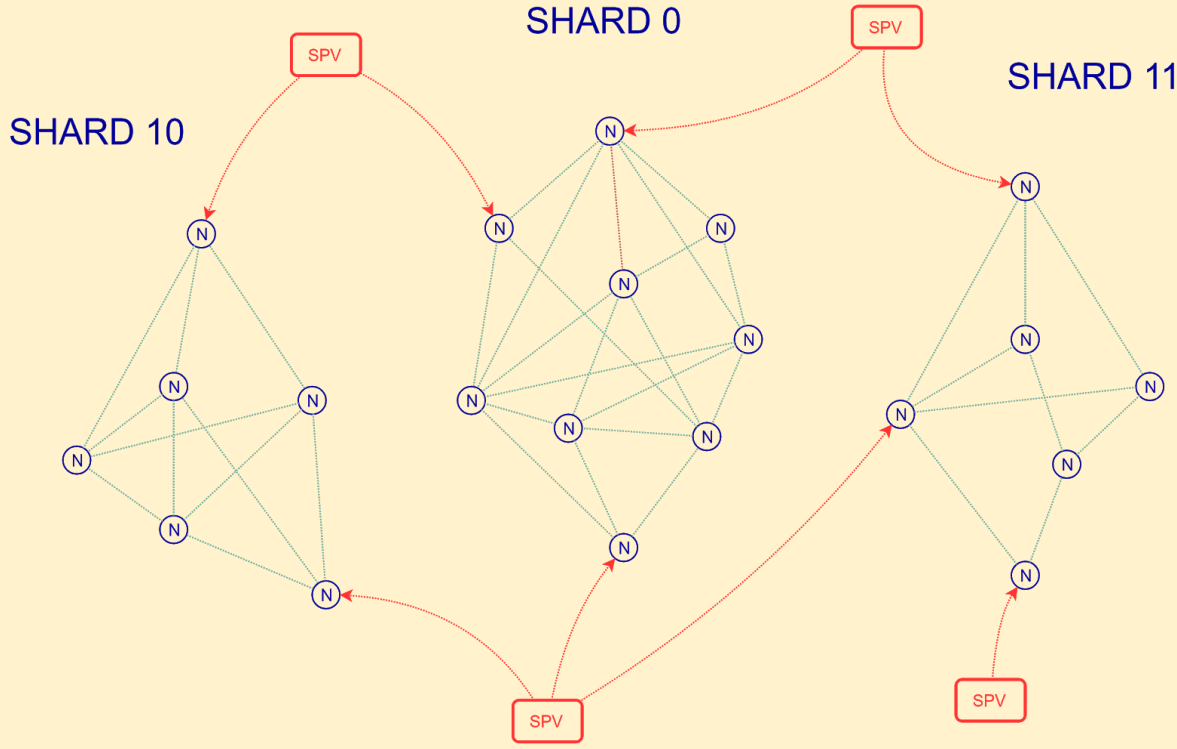
Yggdrasil Sharding Protocol

Yggdrasil shards the Endurio by splitting it into 2 side-chains called high shard and low shard with the following properties:

- Transaction throughputs should be equal between two shards.
- All the transactions with higher value reside in the high shard, the rest stays in the low shard.
- Tokens can be transferred between shards, with normal transaction fees. This will keep the token price the same in all shards.
- Full-client only works on 1 shard at a time. But SPVs and wallets connect to full-clients of as many shards as necessary.
- The block mining reward, difficulty and Service Node requirements will be split proportionally with the total value of each shard's transaction set. The total block reward of all shards is always 1.00 NDR.



The chain will eventually grow into a tree, where each branch (or shard) independently works on a subset of UTXO from its transactions. Each address can have UTXOs in many shards. This removes the responsibility of managing shards off the network, to the user and client software which have the best incentive to secure their own money. See Security for wallet shard management.



Sharding Strategy

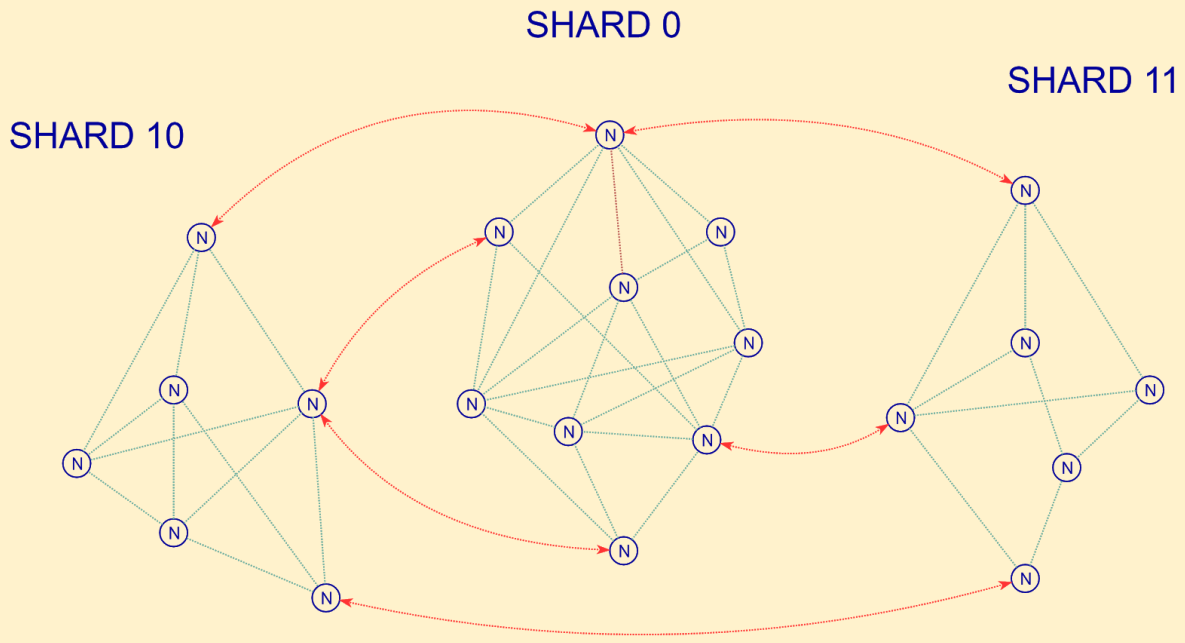
To protect the benefit of both user, Service Nodes and Miners, a shard is split only when a specific threshold of block saturation is reached. Splitting too frequent will create too many unsaturated shards, with transaction fee too low to properly incentivize the Service Nodes network. Splitting too infrequent will result in oversaturated shards with transaction fee too high and drive the user always.

A shard is fully saturated when all the blocks in an epoch (1 whole calendar week) reach maximum block size. Only transactions with non-zero fee count. A shard will be split when it reaches 90% saturation. (This threshold is subject to change in the development process.)

Cross-shard Communication

Once split, (even though it's technically possible,) shards (or branches) will never be merged. Most of the everyday transactions does not require communication between shards, but occasionally some do. Cross-shard communication is performed by one full-client opening a connection to another full-client of a different shard. This connection will provide the state of transactions from other shards, so full-client can perform the following cross-shard operations:

- Cross-shard transfer.
- Cross-shard 2-ways transactions, or swap.



In a non-finality consensus, all states of confirmation can be reverted. Endurio's Lock & Block consensus is PoS/PoW hybrid, which is non-finality, a locked or blocked transactions can always be reverted when the longest chain has a conflict detected. When a cross-shard operation conflict is detected, transactions from higher order shard takes priority, and the conflicted transactions from lower order shard will be reverted, no matter how long the reverted chain is.

In a pure PoW system, reverting a long chain might sound really bad, but with the Lock & Block protocol, only the double-spend transactions would be affected in the event of chain reversal. Non-conflicted transactions in a reverted block will return to the locked state, and will eventually be included in one of the next blocks.

Security

The security of Yggdrasil sharding protocol relies on its economic-driven property. After a shard split, transaction throughput should be equal between the 2 new shards. Let's say, the high shard has 80% of original shard's STB value, and the low shard got 20%. The new high shard will have the following properties:

- 80% of STB value.
- 80% of NDR value.
- 80% of mining reward.
- 80% of mining difficulty.
- 80% of Service Node requirements.

The low shard obviously has everything at 20%. This will naturally split the mining power and Service Node network at exactly the same percentage (80/20), because anything else is economically inefficient for all participants. [TODO: insert math proof here]

This resource split mechanism makes sure that high-value transactions are protected by more mining power and stakes. (Transactions worth of 80,000 USD should be 4 times more secure than 20,000 USD transactions.) This efficiently protects the network from 51% attack and shard-hopping attack, which are major challenges of sharding in public blockchain.

In pure-technical sharding schemes, transactions are usually split randomly, while mining power and minting stake are split evenly between shards. This allows high-value transactions can occur in all shards, while the security level of each shard is divided. Yggdrasil keeps all the high-value transactions in one shard, with higher security, while letting all low-value transactions in the other shard, with less security. Any adversary attempts to attack either shard should face the same cost versus benefit problem. It's easier to attack the lower shard, but also less worthy.

The protocol itself does not force the transaction value limit on each shard. Users can still receive transactions with high-value on low shards (for lower fees or for bad intention). But doing so, they risk their own money getting double-spent or reverted. It is the user's responsibility to only accept high-value payment in high order shard, and reject ones in low order shard. Every wallet applications should perform this check, and alert its user when there's such a suspicious incoming transaction.