Staking Service Network

Serving With Responsibility

PHAM Tuan Anh (Zergity@gmail.com)

DRAFT 2018.08.09

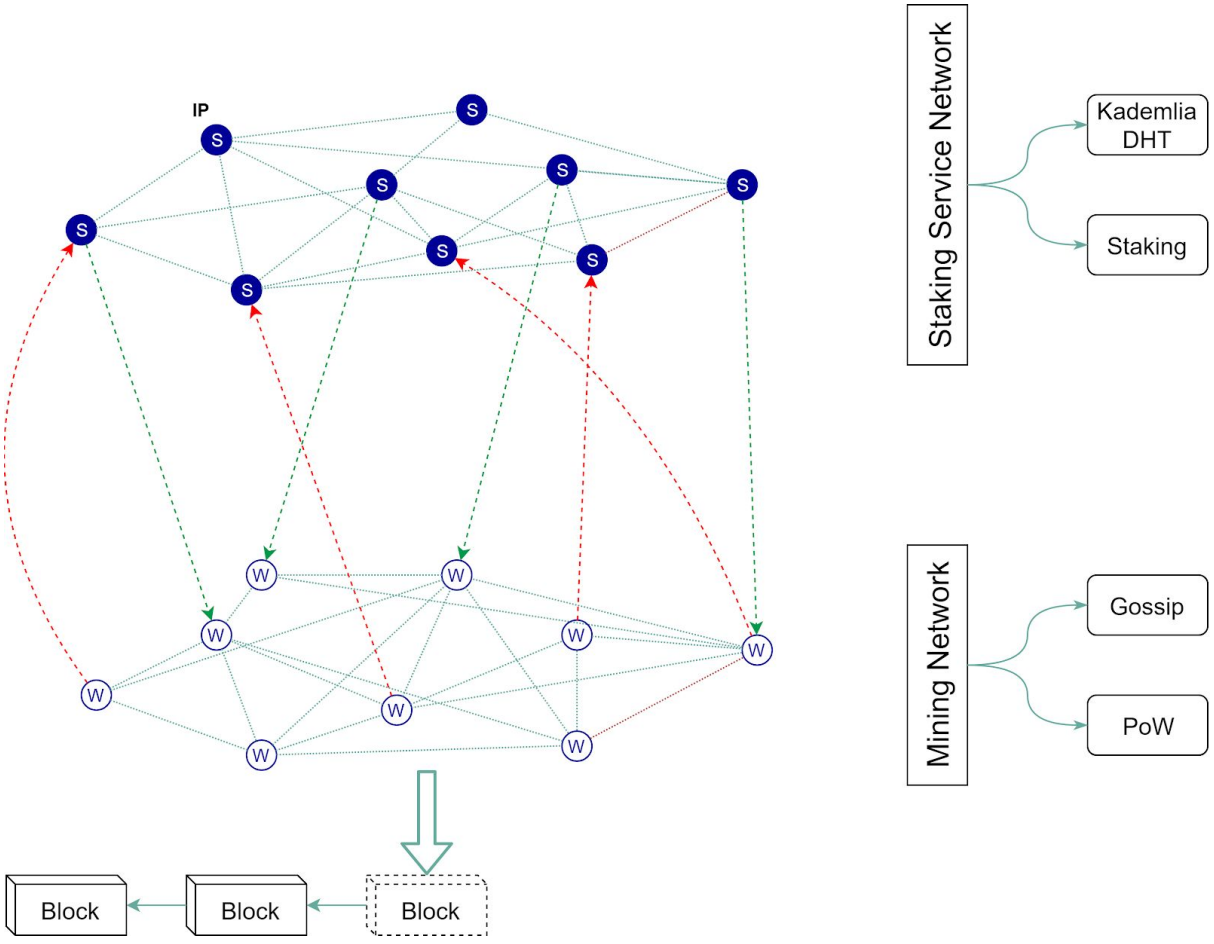
This paper is part of YggChain project.

Table of Contents

The 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.

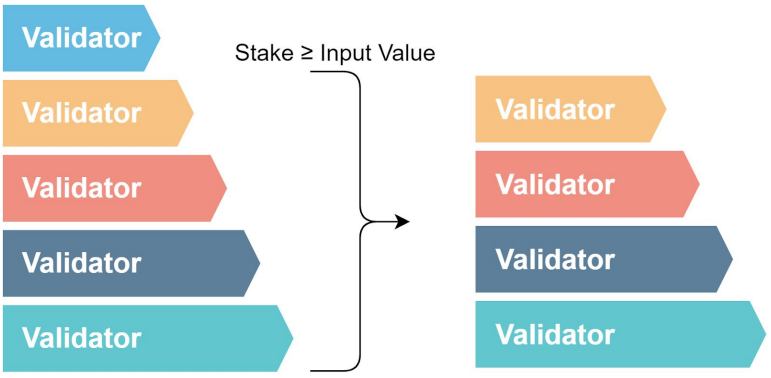
Desirable properties of the SSN:

- 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:

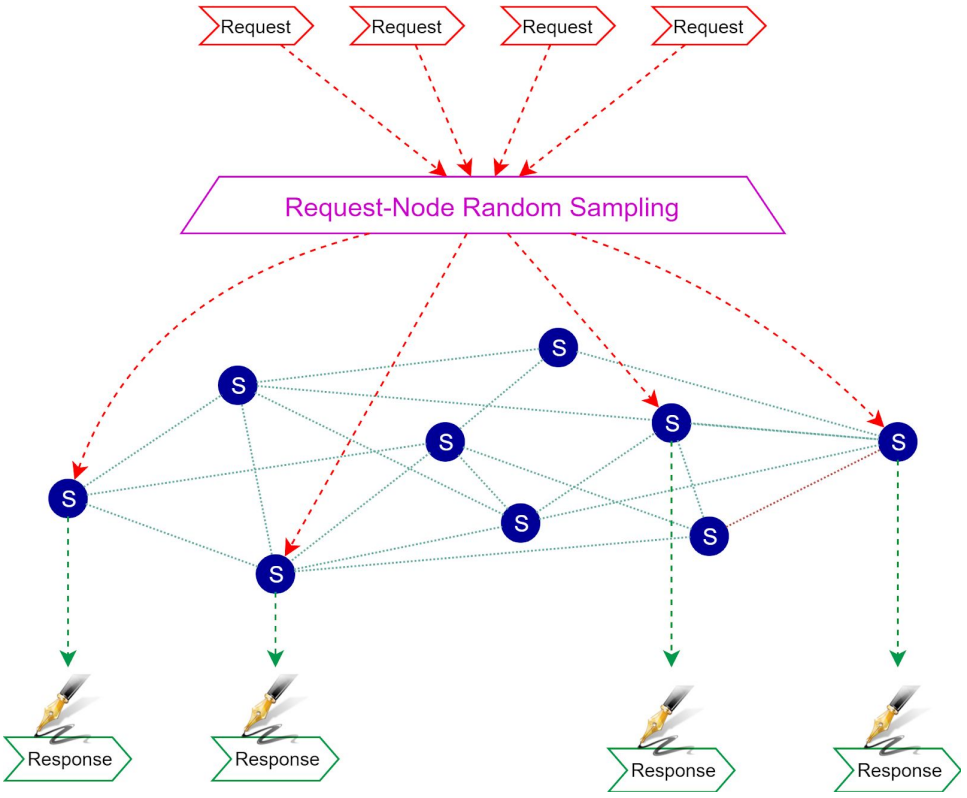
1. Prevent Sybil attack on the SSN.
2. Will be destroyed (or slashed) when a node is detected with bad behaviour.

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.



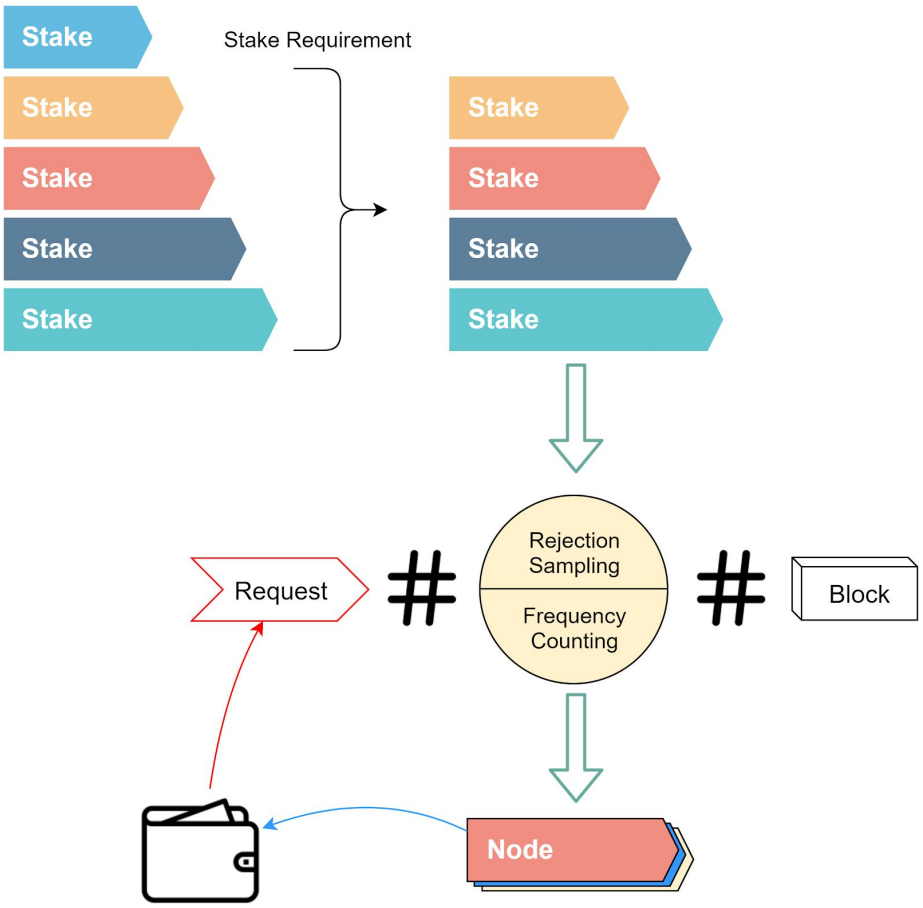
Request-Node Random Sampling

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. Using the last block hash and the request data as entropies.
4. Using Rejection Sampling or Frequency Counting algorithm on the list to select a serving node.

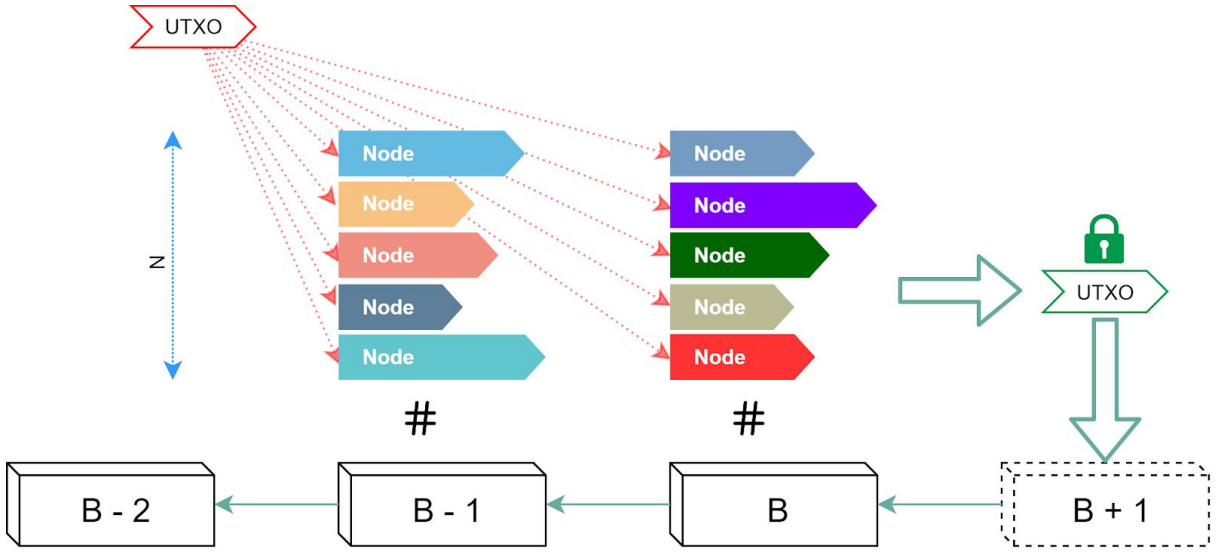
Staking nodes can be offline or maliciously unresponding, so services often select a set of nodes instead of a single one to serve. Larger set provides a higher level of fault tolerant with the cost of higher network footprint.

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.

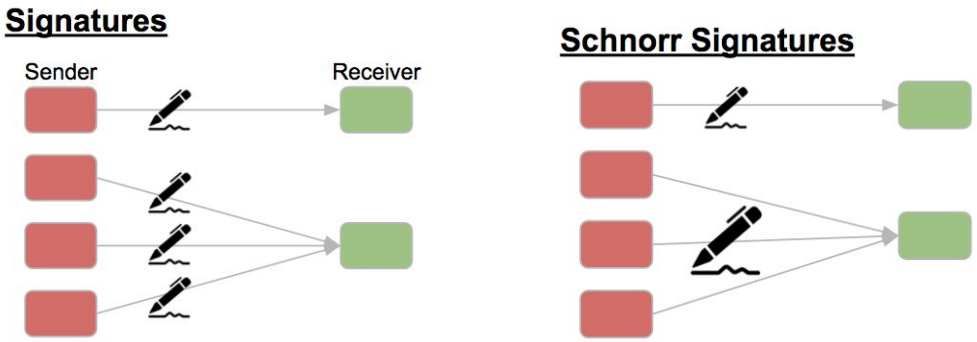


Since service nodes are not always online to serve, and some of them might even be not honest, Input Locking should be Byzantine Fault Tolerant and be working under a certain level of availability. To provide fault tolerance, ILS also maps an UTXO to a set of N nodes instead of one, and the lock is successful 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+3) 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.



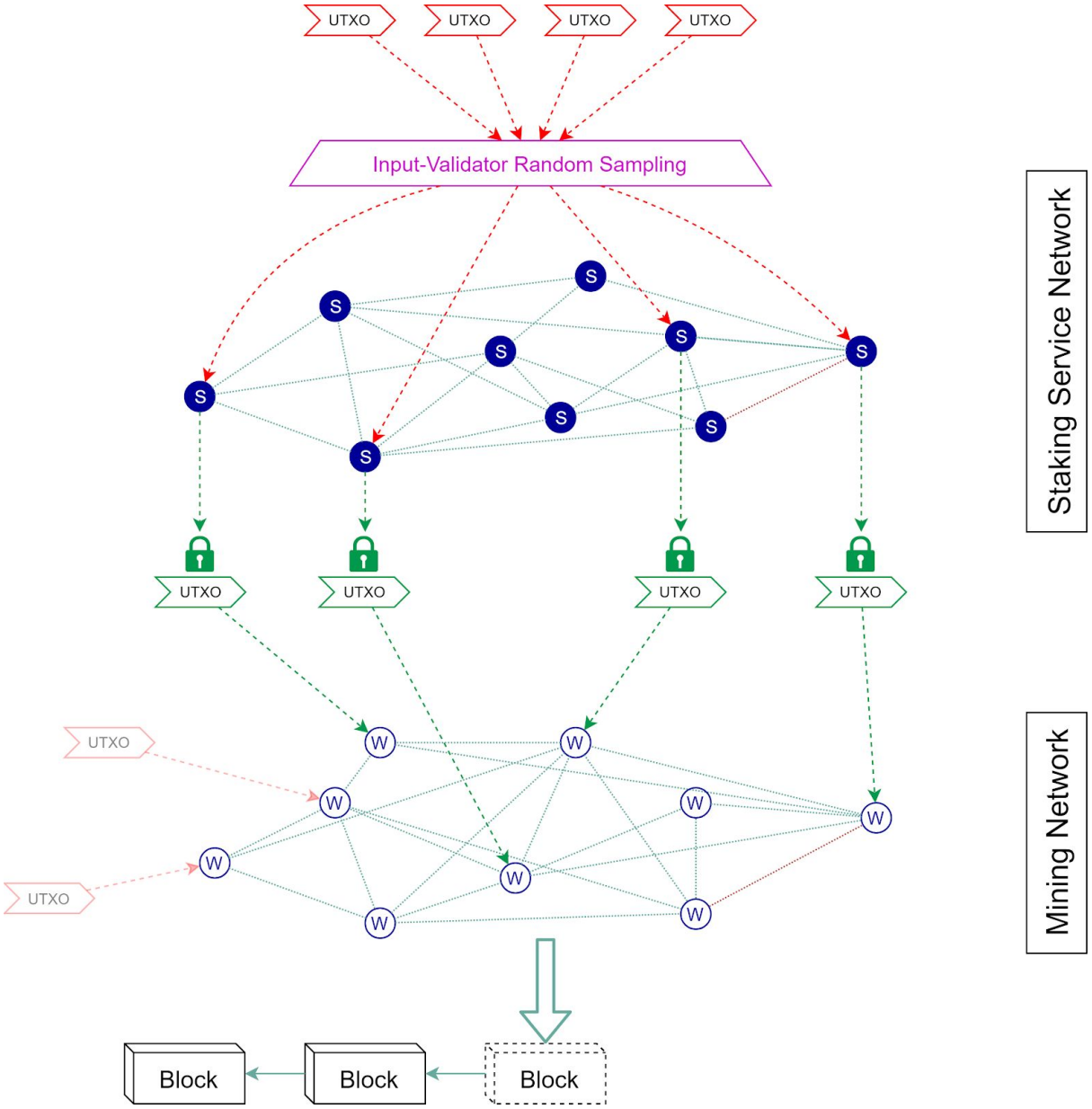
Lock & Block

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 sent 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.