Blockchain components

NeoFS Sidechain Governance

NeoFS uses the sidechain as a database to store meta-information about the network: network map, audit results, containers, key mappings and network settings and several supplementary thins.

The sidechain works on the same principles as the mainnet: there are no free transactions, the committee chooses the consensus nodes, etc. It gives many advantages, for example, limited GAS Utility Token supply prevents some kinds of DoS attacks, and the one can use the same Neo3 mainnet tool stack to work with NeoFS sidechain information.

To effectively work with sidechain we need to solve the following problems:

- How the mainnet committee can control Inner Ring nodes and the consensus nodes of the sidechain?
- How Storage Nodes and Inner Ring nodes get sidechain GAS Utility Token to send transactions?

NeoFS Governance model solves these problems with seven "Alphabet" sidechain contracts and first seven Inner Ring nodes bound to those contracts, acting as the sidechain committee.

Alphabet contracts

Alphabet contracts are seven smart contracts deployed in the sidechain, named after the first seven Glagolitic[12] script letters: Az(⊠), Buky(⊠), Vedi(⊠), Glagoli(⊠), Dobro(⊠), Jest(⊠), Zhivete(⊠). These contracts hold 100,000,000 sidechain NEO Token on their accounts (approximately 14,285,000 for each). By storing NEO Token on the contract accounts, we protect it from unauthorized use by malicious sidechain nodes. Contracts do not transfer NEO and use it to vote for sidechain Validator nodes and to emit GAS Utility Token.

Alphabet Inner Ring nodes

Alphabet Inner Ring nodes are the first seven nodes in the Inner Ring list that are logically bound with one-to-one relation to Alphabet contracts. They are the voting nodes making all decisions in the NeoFS network. All other Inner Ring nodes take care of Data Audit, Storage Node attributes verification and other technical tasks.

Being Alphabet node implies running the sidechain Consensus Node using the same key pair as the NeoFS Inner Ring node instance. Hence, Alphabet node candidate must:

[12] https://en.wikipedia.org/wiki/Glagolitic_script

- Setup NeoFS Inner Ring node instance
- Setup NeoFS sidechain full node using same key pair
- Register the same key in mainnet NeoFS Inner Ring candidates list
- Register the same key in sidechain committee candidates list

Alphabet contracts invocation

Contracts cannot distribute utility token or vote by themselves. To do these operations, Inner Ring nodes invoke alphabet contract methods. Alphabet Inner Ring nodes can invoke only corresponding alphabetical contracts. One node invokes one contract.

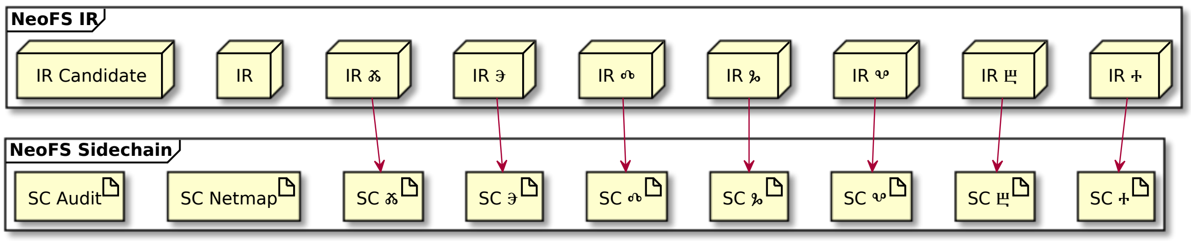

Figure 3: Inner Ring to Alphabet SC relation

Alphabetic contracts have hardcoded indexes. Contracts authenticate method invoker by using the list of Inner Ring node keys from Netmap Smart Contract. This scheme helps to limit malicious Alphabet Inner Ring node actions and make network more resiliant to Inner Ring nodes losses.

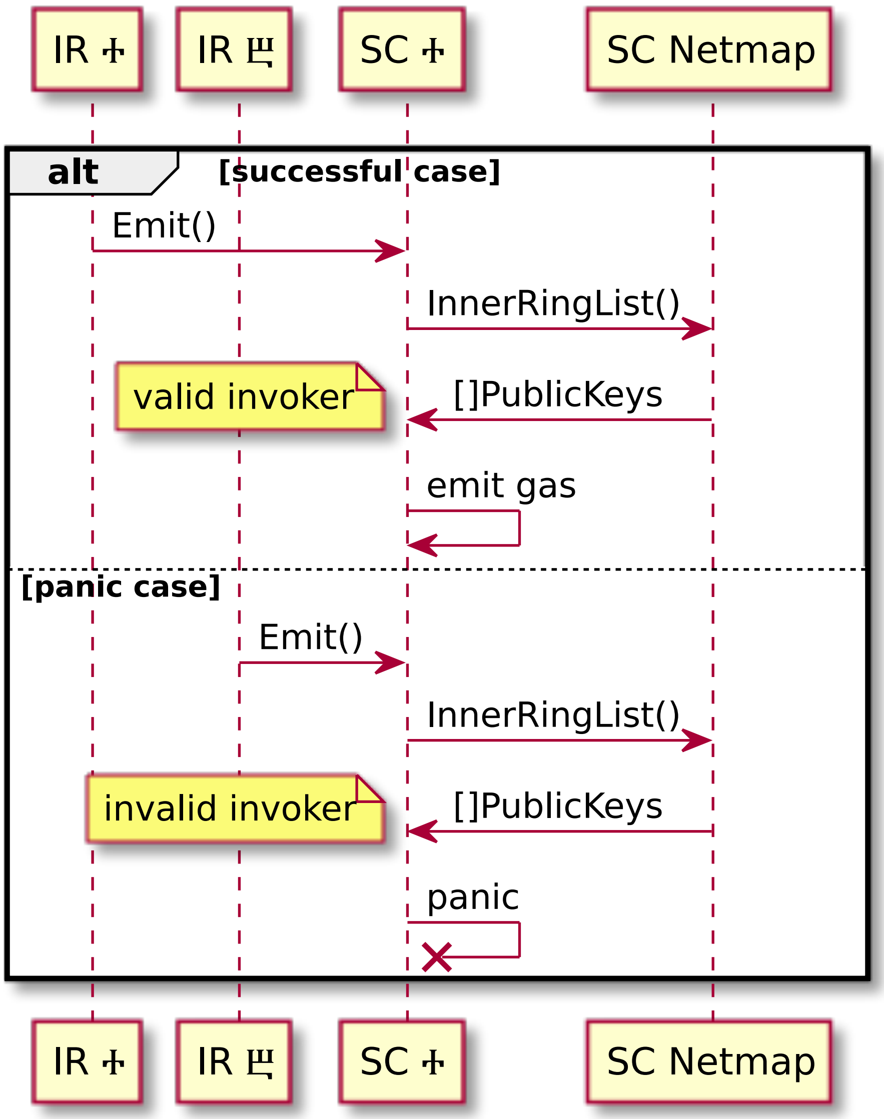

Figure 4: Alphabet SC invocation by Inner Ring nodes

Utility token distribution

Inner Ring nodes invoke `Emit()` method in corresponding alphabetical contracts. This method transfers all it's NEO Token to it's account, thereby producing utility token emission. Within the same invocation context, the contract transfers a share of the available GAS Utility Token to all Inner Ring wallets. Contract will keep the $\frac{1}{8}$'th part on it's balance as an emergency reserve.

$$InnerRingNodeEmission = G \cdot \frac{7}{8} \cdot \frac{1}{N}$$

G - contract's GAS Utility Token amount
N - length of Inner Ring list

After receiving GAS Utility Token, the nodes of the Inner Ring can periodically transfer a share to all registered Storage nodes and use the received utility token for sidechain operations: change epochs, register new containers, save data audit results, etc.

Storage nodes have limited supply of GAS Utility Token to prevent malicious actions and DoS attacks. Depending on Storage Node activity and reputation records it will get different utility token amount, normally enough to perform all required operations. Sidechain GAS and mainnet GAS are different tokens, hence Storage Nodes don't spend rewards for operations and can't convert sidechain utility token into mainnet GAS Utility Token or vice versa.

Changing sidechain validators

Beforehand, Alphabet Inner Ring node candidates register validator keys in the list of candidates for the sidechain committee. When sidechain Netmap smart contract send notification on Inner Ring nodes list updates, Alphabet Inner Ring nodes invoke `Vote([]keys)` method of all Alphabet smart contracts to gather signatures and then make them vote for sidechain Committee. Each Alphabet contract votes for the keys proposed by sending `VotesPerKey` votes for each key. Normally, there is just one key per node, hence N equals `1`.

$$VotesPerKey = \frac{A}{N}$$

A - contract's NEO amount
N - length of proposed keys list

Changing the Inner Ring list

Inner Ring nodes follow a self regulation process, allowing them to vote for substitution for dead or malfunctioning nodes with new ones from the candidate list. Only Alphabet nodes prepare new Inner Ring nodes lists and vote for it, but all nodes listed there must confirm their participation via the same voting mechanism.

The voting procedure uses sidechain `Voting` smart contract, but the list of candidates is taken from mainnet NeoFS contract. When Inner Ring nodes agree on the updated list, it's submitted to mainnet NeoFS smart contract and then mirrored back to Netmap smart contract on sidechain.

By using `Emit()` and `Vote()` methods of Alphabetic smart contracts, Inner Ring nodes take full control of the sidechain. They control validator keys and utility token distribution. Thus, if the mainnet committee will control list of Inner Ring nodes, then it will control sidechain as well.

The mainnet Committee can set the list of priority candidates for Alphabet Inner Ring nodes in mainnet `DesignationContract`. Nodes from that list will be voted for becoming Alphabet Inner Ring nodes and substitute current Alphabet nodes, if they confirm the following requirements:

- Node's key is registered as a candidate in mainnet NeoFS smart contract
- Node's key is registered as a sidechain committee candidate
- Node is not listed as inactive in sidechain `Netmap` contract

The `DesignationContract` list may contain any number of valid candidates, the voting process will make sure as many of them, as possible, are in the first seven active inner Ring nodes. If there is not enough appropriate candidates, the rest will be taken from the regular candidates list. If there are too many, only the first seven suitable nodes will be used.

The voting algorithm is the same for each Inner Ring node and starts in the following cases:

- New Epoch
- Notification from mainnet DesignationContract on Inner Ring nodes list change
- Notification from sidechain Netmap contract on inactive Inner Ring nodes list change

All Inner Ring nodes listen for notifications from Voting contract and if they see themselves in the new Inner Ring nodes list, they confirm their participation by sending the same list in `Prepare()` method. Only newly added nodes need to confirm their participation with a transaction. If the node is already in the active Inner Ring list, it doesn't need to send confirmation.

When there are enough Alphabet signatures and all required candidate signatures sent with `Prepare()` method, the last invocation will update the list and finish voting round.

Active Alphabet Inner Ring nodes will be waiting for the round to end and locally test invoke `EndRound()` method. When voting round timeout comes and round is not finished succesfully by agreeing on a new list, one of the Alphabet nodes will invoke `EndRound()` and settle the round results.

If by the end of the voting round some newly added nodes didn't confirm their participation, they are added to Netmap smart contract's inactive list. This will trigger a new voting round without those inactive nodes.

If there are not enough candidates, Inner Ring nodes will accept the best list they can gather.

When the new list is agreed, `Voting` smart contract sends notification. All Alphabet nodes react with invocation of `UpdateInnerRing()` of mainnet NeoFS smart contract. When the majority of Alphabet nodes send the update and mainnet list is updated, it will be mirrored by Alphabet Inner Ring nodes in sidechain `Netmap` smart contract.

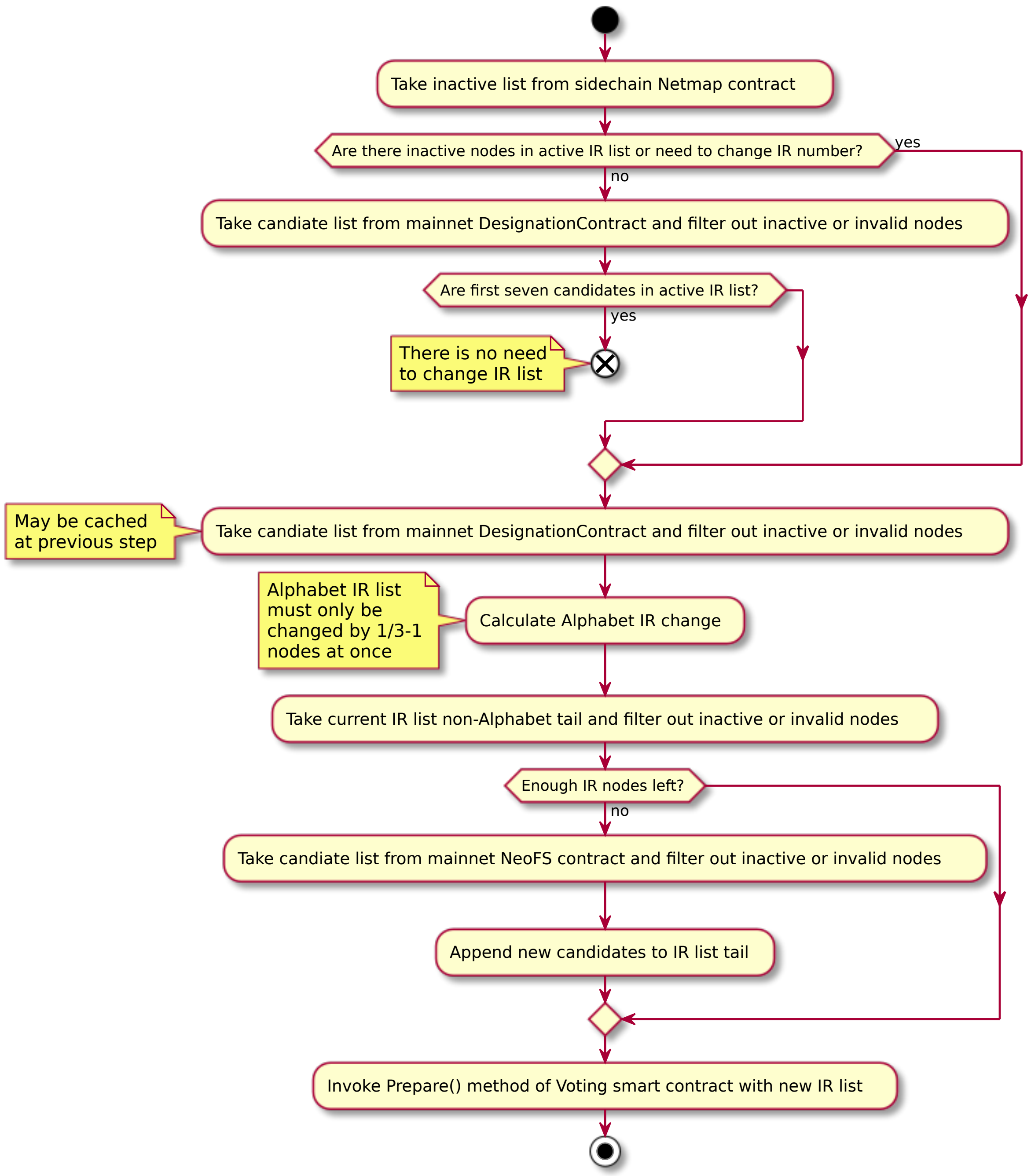
Take inactive list from sidechain Netmap contract
Are there inactive nodes in active IR list or need to change IR number?
yes
no
Take candiate list from mainnet DesignationContract and filter out inactive or invalid nodes
Are first seven candidates in active IR list?
yes
There is no need to change IR list
May be cached at previous step
Take candiate list from mainnet DesignationContract and filter out inactive or invalid nodes
Alphabet IR list must only be changed by 1/3-1 nodes at once
Calculate Alphabet IR change
Take current IR list non-Alphabet tail and filter out inactive or invalid nodes
Enough IR nodes left?
no
Take candiate list from mainnet NeoFS contract and filter out inactive or invalid nodes
Append new candidates to IR list tail
Invoke Prepare() method of Voting smart contract with new IR list

Figure 5: Inner Ring Alphabet node voting algorithm

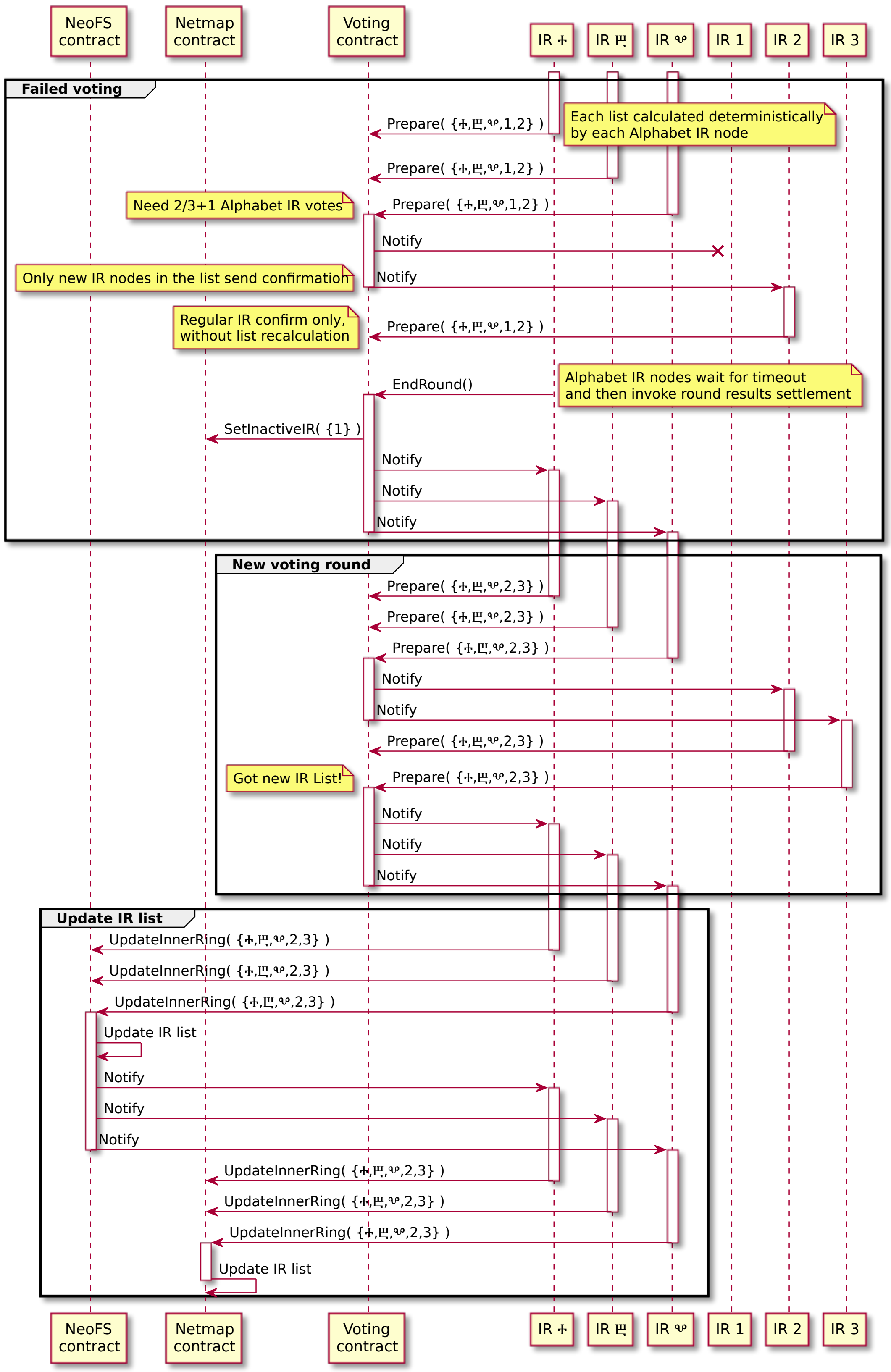

Figure 6: Inner Ring list update in mainnet and sidechain