EchoRand

The following document describes the basic working mechanism of **EchoRand** consensus algorithm which underlies the Echo blockchain network.

The basis for the **EchoRand** algorithm is the Algorand v9 theoretical work, which describes reaching a consensus in a decentralized network based on the solution of the Byzantine generals problem. In Algorand v9 work, several possible solutions of the algorithm are presented. A solution called **Algorand'2** is taken as the **EchoRand** basis.

Namely:

- The way to determine participation of a specific party in each particular step is changed
- Disposable, derived keys for network messages signing are renounced.
- The way of generating a shared random state in the third step of the BBA algorithm is changed.

Basic components

The following components underlie EchoRand:

- **executor** - the network account selected in the step of the round for performing a specific consensus action
- **set of producers** - an identified by the protocol for the current block list of participants, that are given the opportunity to propose a block option for the current round
- **set of verifiers** - an identified by the protocol for a specific step list of participants, that are trusted to perform the verification actions which are defined by the step, in which they were chosen.
- **verified random function (VRF)** - a pseudo-random function, which provides publicly verifiable evidence of its conclusion.
- **round seed** - a pseudo-random value changed on each block. It serves as the basis for generating the verifiers set and block producers.
- **Graded Consensus** - one of the consensus stages, at which each verifier must announce their preliminary block determination regarding the current block

- **Binary Byzantine Agreement (BBA)** - a solution of the Byzantine Agreement problem, which has transfer of binary data between participants and reconciliation of results with the overall picture in its basis.
- **a network node** - a running echo application instance, located in the network of other nodes, having the last actual state (synchronization to the last block)

Legend

Designation	Description
$r >= 1$	the current round of the algorithm, it is actually equal to the number of blocks in the base plus 1
$s >= 1$	the current number of the algorithm step in the round
Q_r	**r** round seed
$VRF(r, s)$	an ordered plurality of executors who participate in **s** step of **r** round
$VRFN(r, s)$	an ordered indexes plurality of executors indexes from $VRF(r, s)$, who are registered at the current node and participate in **s** step of the **r** round

VRF

The concept of verifiable random function (VRF) was introduced by Mikali, Rabin and Wadhan. This is a pseudo-random function that provides publicly verifiable evidence for the correctness of its conclusion. For a given input value `x`, the owner of the secret key`SK` can calculate the value of the function $y = F_{SK}(x)$ and the proof $P_{SK}(x)$. Using the proof and public key $PK = g^{SK}$, everyone can verify that the value of $y = F_{SK}(x)$ is indeed calculated correctly, but this information cannot be used to search for the secret key.

The use of VRF in EchoRand is as follows - having a pseudo-random value for each Q_r round and the VRF function, each of the network nodes can determine the list of $VRF(r, s)$ executors in **s** step of **r** round, and based on it, perform the necessary actions if the authorized account on the node is part of $VRF(r, s)$, and as well verify whether the participants have the right to act at this step.

Definitions of Active Performers

The checked random function at each **r** round and **s** step is built iteratively, as follows:

$$1.VRF_0(r,s) = SHA256(Q_{r-1}, r, s)$$
$$2.VRF_n(r,s) = SHA256(VRF_{n-1}(r,s))$$

The result of this function is an array of random values:

$$VRF(r,s) = VRF_0(r,s), VRF_1(r,s), ...$$

A specific executor is calculated from the $VRF_i(r,s)$ hash in such a way, that the probability of the choice of the participant as active, is proportional to his balance in the system at the time of the `r - 2` block.

$VRFN(r,s)$ set is an array of indexes that is different for each node of the network, and if $i \in VRFN(r,s)$, then the user ID is calculated from $VRF_i(r,s)$ that is the executor for the given round and step at the selected node.

In other words, $VRFN$ is a selection of those executors from VRF, who are authorized at the current node and must be executed on a specific round and step.

At different network nodes, at the same round and step of the algorithm, the $VRFN$ pluralities will be different, and the VRF plurality will be the same.

Round Random Value - Round Seed - Generation

The $Q(0)$ initial vector is randomly selected while the database is initialized.

Further, the Q_r vector is calculated as follows while creating a new block:

For a non-empty block $B(R)$:

$$Q_r = H(signQ_{r-1}, r)$$

In this case, the signature uses the private key of the participant that creates the block.

In case $B(R)$ block is empty:

$$Q_r = H(Q_{r-1}, r)$$

Generating a random value at s = 7,10,13, … BBA step

$$BBARAND(s) = lsb(SHA256(Q_{r-1}, r))$$

Consensus rounds

The main stages of consensus:

- Block generation by performers selected to be the creators of the block
- Voting for the best block
- Reaching an agreement between nodes about the best unit

Block generation

A certain number of accounts (a block creators set) generates a block.

The block creators set is determined by each block with the help of verifiable random function (VRF). As a result, each network node receives a $VRF(r, s)$ set and a $VRFN(r, s)$ subset - a list of accounts authorized at this node. If $VRFN(r, s)$ is not empty, the node issues a block proposal based on the transactions that are in the node mempool.

Voting for the best block (Graded consensus)

It consists of 3 steps. At this stage, the goal of the verifiers is to vote and announce to the network, which of the producers they consider to be the best candidate for the current block.

Step 1 - the voting

Each of the selected verifiers tells the network which of the blocks they consider preferable for the current round.

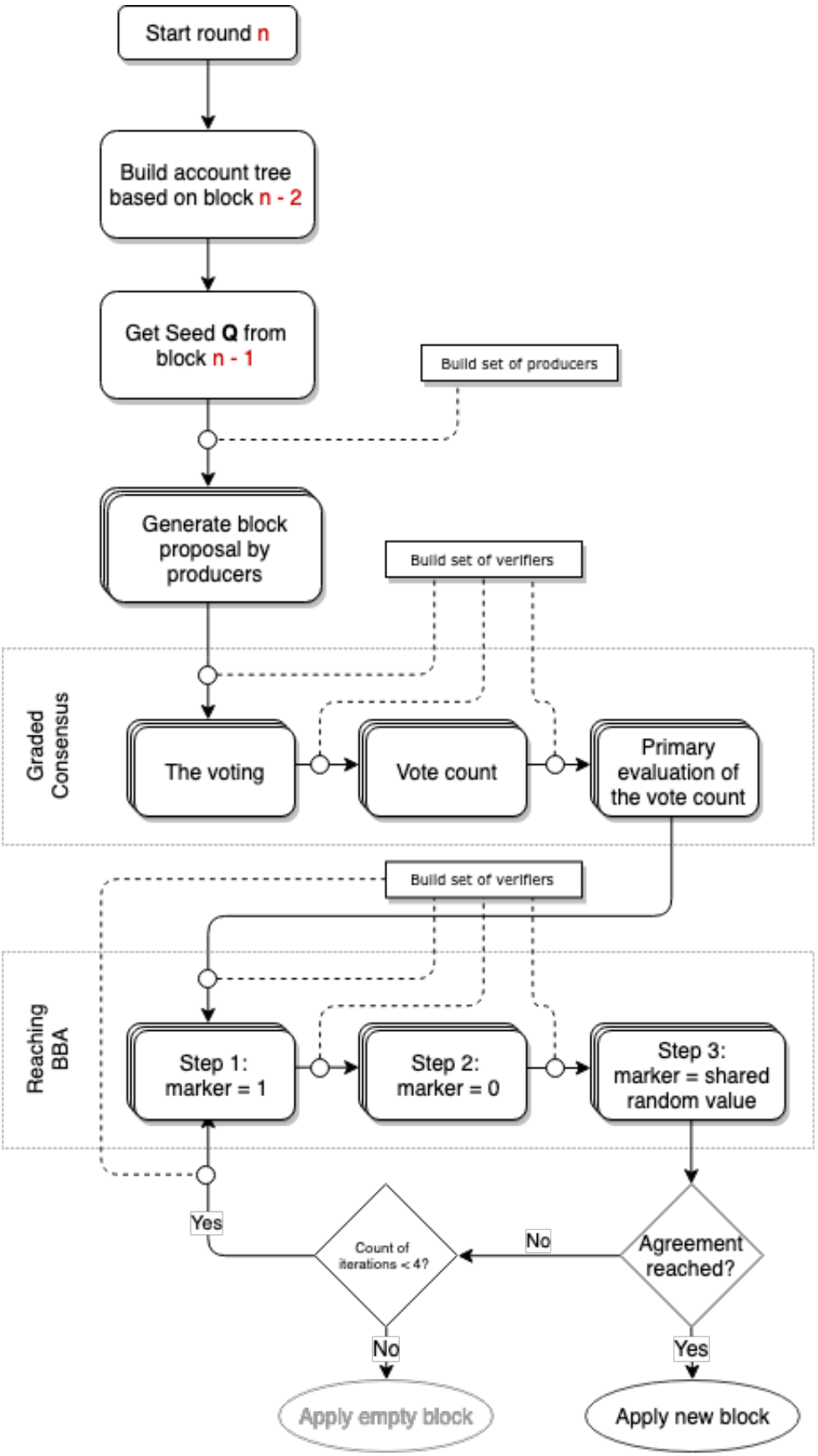

Figure 1: echogrand-steps.png

Step 2 - vote count

Based on the messages received from step 1, to determine which of the producers got more votes and to announce it to the network. (it is done by each of the verifiers).

Step 3 - primary evaluation of the vote count

Having the voting results of the previous steps, the nodes know, whether the network was able to agree on the choice of the producer for the current round. Each verifier forms a message including information on the result(agreed or not) and what exactly they have agreed on, and sends the message to the network.

After this step, all nodes in the network have a preliminary idea of whether the producer of the unit has been determined or not. In an honest network, this would be enough to complete the round. But since we allow the possibility of unscrupulous participants, the network needs to verify the data. This is the objective of the next stage.

Reaching Binary Byzantine agreement (BBA)

At each step of the algorithm work, the nodes in the network can be divided into two pluralities:

- nodes that have received a sufficient number of messages in the previous round(s) (with some identical value), allowing them to offer this value as a solution.
- nodes that have received two solutions in messages and can't give preference to any of them.

In the latter case, undecided nodes use VRF to generate a shared random number from the plurality {0, 1} to make a decision and to send it. Since the random number will be the same for all "unsure" nodes, all such nodes will take identical decision.

The stage consists of cycles, which include 3 steps each. At each step, a new set of verifiers sends their vision of the vote result vote in binary form. If as a result of the cycle (3 steps) 2/3 + 1 verifiers agree, the block is applied. If not, the cycle starts again.

If in 4 cycles (12 different sets of verifiers participate in them) the network didn't manage to come to a common opinion, an empty block is used in the network and the next round starts from the very first step - block generation.

Block application by the network participants

All network nodes receive all messages sent by producers and verifiers at all stages of the consensus. Accordingly, each of their nodes at the time of the consensus ending determines its end for itself, and understands which block to apply and add to the chain. It means, the final message with the resulting information isn't sent by anyone, as it's not necessary.

Branching permission

The number of steps of the algorithm and dependence on the whole accounts database makes the possibility of branching unlikely. However, EchoRand still has a branch permission mechanism. The branching permission takes place according to one of the following scenarios:

To switch to the longest chain in the presence of several chains.

- If there is more than one long chain, to follow the one, in which the last block is not empty. If all of them have empty blocks in the end, to check the second and subsequent blocks from the end to the first non-empty block.
- If there is more than one long chain with non-empty blocks in the end of a R-length chain, to follow the one in which the r block has the smallest hash value.

Network protocol optimization

To reduce the number of messages with information about the block proposal, the following optimizations are implemented in the protocol:

- if the network node receives a block proposal that is not the first for the round and is not better than the previous one, the node does not send a message about it to the other nodes.

- If several participants are authorized on the node for the block generation round, the node itself determines which of the blocks is the best candidate and sends a network proposal only to one block.

Exceptional situations

No network

Steps of the algorithm will not receive messages and all exits from the steps will occur only when the timer is triggered.

Since the end of the round at the moment occurs only as a reaction to an incoming message, the BBA steps will be executed in a loop until reaching the μ constant. As a result, an empty block will be generated.

Network Restoring

Nodes that as a result of network recovery will enter the round in the middle, will contain incomplete data in their round contexts. As a result, they will either generate incorrect estimates or vote for an empty block.

In each of these options, the nodes will behave as node-intruder. As a result, information from such nodes will be filtered by the BBA algorithm. Due to incomplete data in local contexts, such nodes will complete the round:

- with a wrong block
- with an empty block

A branching, which will be automatically resolved when the rest of the network goes ahead in the process of generating new blocks, will occur.

An incomplete blockchain base in node

It's the case in which the local database of the node is catching up with the network database. In this case, the algorithm cannot work due to the fact that there are no values:

- HB_{r-1} - is hash of the last created block

- Q_{r-1} - is a random value of the last round of the algorithm

It is required to determine the moment when the local database will "catch up" with the network database and launch a round of the algorithm.

Lack of active performers in the step

A participants set makes calculations using VRF and does not depend on the actual presence of participants in the network. There may be a situation when for some step of the algorithm there are no active performers. In this case, the active members of the network at the expiration of the timeout simply go to the next step or use an empty block, in case it was the last step.

Delegation of participation in consensus

The participants on the rounds are the accounts, but to participate in the consensus, the active node is needed, since only having the current network status and the availability of free transactions in the mempool allows one to determine the sets of the performers, to collect and check messages.

Given that most accounts do not have the ability to maintain an active node in the network, but can be selected to participate in the round, the protocol implements the mechanism for delegating participation in consensus to other accounts. This means that an `A` account can set for itself a trusted `B` account with a knowingly running node in the network and thereby give the `B` account an opportunity to issue consensus messages at the moment when the `A` account was selected by the participant.

By default, the `B` trusted account for the `A` account becomes the account that registered the `B` account in the network.

> Creating an account in the Echo network requires creating a corresponding transaction added to the block