

Proof-of-Rounds

PoR

The next generation Consensus Algorithm

Reconcile high speed/scalability & high security

WHITEPAPER

Revision 2018/10/23 SINCE 2018/05/01 early works co.,LTD

Contents

1. Overview

1.1. What is Proof-of-Rounds?.....	5
1.2. Approval of transactions by permutations.....	6
1.3. Optimization of processes by assigning roles using a node Hierarchy.....	7
1.3.1. Streamlining of information distribution using a Hierarchy.....	8
1.4. Construction of a unique network that distributes information efficiently.....	10

2. Scalability

2.1. The scalability problem.....	13
2.2. DAGs.....	14
2.2.1. Existing DAG systems.....	14
2.3. DAGs in PoR.....	15
2.4. Benefits of PoR for DAGs.....	16
2.5. Optimization of network structure.....	17
2.5.1. Round network.....	18

3. High Speed

3.1. PoR approval speed.....	21
3.2. Assignment of roles to each node.....	22
3.3. Normal nodes.....	23
3.4. Super nodes.....	24
3.5. Verification of transactions by nodes.....	25
3.6. Master nodes.....	27
3.6.1. Connections between master nodes.....	27
3.6.2. Appending to the round header.....	28
3.7. Root nodes.....	30
3.8. Synchronization between nodes using signals.....	31
3.9. Elimination of nonces.....	33
3.10. Speed summary.....	36

4. High Security

4.1. Security.....	38
4.2. Hash functions.....	39
4.3. Digital signatures.....	40
4.4. Elliptical curve cryptography.....	42
4.5. Rigorous consistency checks by each node.....	43
4.6. Unspecifiability of signal nodes.....	45
4.7. Management of signal nodes.....	46
4.8. Network reconstruction.....	46
4.9. Resistance to attacks on nodes.....	47
4.10. Elimination of malicious hacking based on economic rationality of nodes.....	48
4.11. Security summary.....	49

1. Overview

1.1. What is proof-of-Rounds?

“A new consensus-building algorithm that reconciles high speed/Scalability with high security”

“PoR” has the following characteristics in order to achieve this.

- **Approval of transactions by permutations**
- **Optimization of processes by assigning roles using a node hierarchy**
- **Construction of a unique network that distributes information efficiently**

1.2. Approving transactions through permutations

The process for approving and appending new blocks to an existing blockchain is established by a consensus-building algorithm such as PoW, PoS, PoC, or PoA and executed through nodes that calculate the hash value based on established conditions.

In contrast, the new consensus-building algorithm known as PoR establishes which nodes will approve transactions in advance, and the approval sequence of these node groups awaiting approval is decided in order of priority. Since each node approves transactions in this sequence, this completely eliminates the need to calculate hash values and removes the restriction of time needed to randomly choose, which has greatly improved processing speed.

The order of priority that determines the sequence of these approval nodes is calculated using the following factors:

- Node age
- Number of successful transaction approvals

These two factors can be calculated by referring to past blockchain data. In this way, it is possible for any node to maintain impartiality regarding sequence because information is mutually distributed.

However, it is not hard to imagine that a major improvement in approval speed is a trade-off with security, and this is in fact true.

For example, the permutation of approval nodes can result in the approval nodes being specified, allowing malicious approvals. Furthermore, the elimination of nonces eliminates complicated operations, which allows transactions to be generated conveniently and manipulated more easily.

This disadvantage is rectified by introducing a “node hierarchy,” which is explained in the next section.

1.3. Optimization of processes by assigning roles using a node hierarchy

A node hierarchy is a system that optimizes the processes of an overall system by dividing nodes into several types and assigning them different roles. The higher the order of the node, the fewer that are installed and the stricter the conditions imposed for their installation.

PoR is composed of the following nodes.

Node name	Node Role
Root Node (RN)	Transaction Consent Append to block chain
Master Node (MN)	Distribution of transactions to SuperNodes Confirm transaction consistency Monitoring the RootNode and other MasterNodes
Super Node (SN)	Confirm transaction consistency Distribution of transactions to each node Discard unnecessary transaction
Normal Node (NN)	Create transaction

Fig, 1.3. Nodes composing a PoR network

As opposed to a system in which transactions can be approved by any node, in PoR, only the RN (root node), which has the highest order in the node hierarchy, is authorized to approve transactions.

An influx of malicious nodes can be prevented by only allowing nodes that have cleared a high hurdle to give approval.

It is also worth noting that the system is such that MNs with established permutations are switched out as RNs sequentially rather than having one node that constantly serves as the RN. This avoids dispersion of the RN's authority and concentrated attacks on the RN, and keeping the RN constantly monitored by MNs who are currently not taking their turn as RN allows for

the automatic elimination of RNs that approve abnormal transactions that could cause inconsistencies.

This monitoring system ensures that, even if there were to be a successful attack on an RN, fraudulent transactions would not be approved and only legitimate transactions could be approved.

1.3.1 Streamlining of information distribution using a hierarchy

MNs also serve to properly distribute information on approved transactions to low-order SNs.

There is a maximum of five SNs for each MN, and these five SNs distribute information simultaneously. The SNs distribute information to low-order NNs, and the number of NNs per SN fluctuates according to network structure.

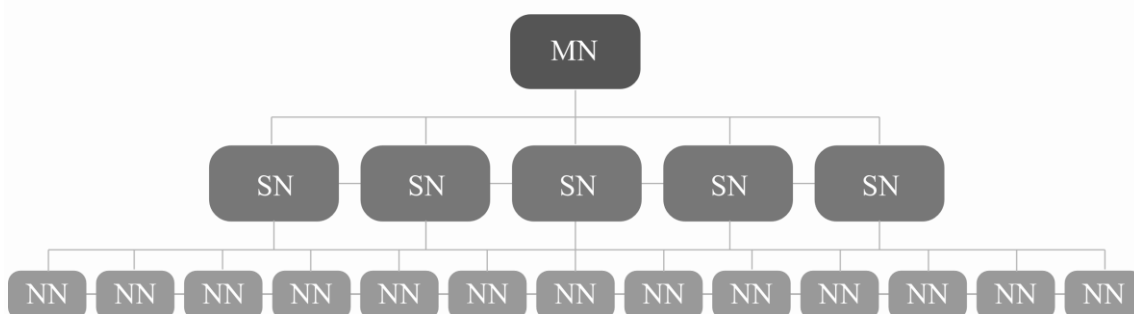


Fig. 1.3.1. Hierarchical structure

SNs serve to reduce the burden of MNs and efficiently distribute information to the network as a whole by supporting the distribution of information to each node through MNs.

Due to this hierarchical structure, the number of MNs and SNs expands and shrinks automatically based on the number of NNs, transaction approvals, etc., and there is flexible scalability that allows free construction based on network structure size.

Since an extremely large number of processes are involved in the matching of transactions sent by NNs, these transactions are first received by the SN and distributed to the MNs once the matching process is

completed. This reduces the load on MNs and achieves rapid information distribution by optimizing processing.

Furthermore, in order to allow the above node hierarchy structure to function more efficiently, PoR uses the new network structure described in section 1.4..

1.4. Construction of a unique network that distributes information efficiently

As described in section 1.3, PoR approves transactions based on permutations. In addition, transactions can only be approved by MNs that have cleared a high hurdle.

In order to efficiently distribute information to MNs with the authority to approve transactions, PoR adopts a circular network structure.

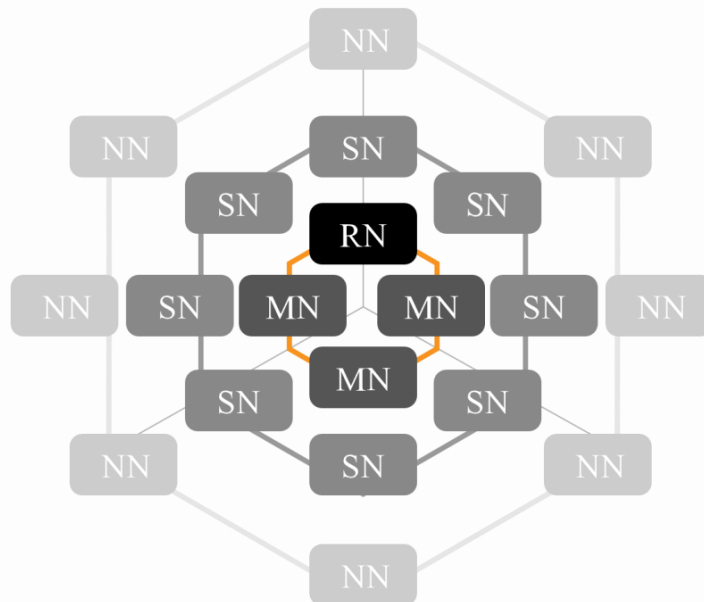


Fig. 1.4. PoR network schematic diagram

SNs and NNs are placed in a circle around the root node and master nodes.

This shape is the origin of the name “Proof-of-Rounds” and is the key factor behind the system's high speed and high security. The information distribution process is as follows.

- (1) The NNs send the transaction data.
- (2) The SNs receive the data, search for matching destinations, and send the data to the MNs.
- (3) The MNs verify the consistency of the information and send it to the RNs if there are no problems.

- (4) The RNs approve the transactions.
- (5) Data approved by the RNs is sent to the MNs at the matching destinations.
- (6) The MNs check the data's consistency and send it to the SNs.
- (7) The SNs send received data to the NNs in a batch.
- (8) The NNs receive the data and the information distribution is complete.

A notable trait of this network structure is the guarantee of impartiality in information distribution.

With a P2P network structure, information is usually distributed to neighboring networks in sequential order. For this reason, as the number of nodes composing the network increases, information is distributed more frequently and it takes longer to complete the distribution. Furthermore, based on the relative positions of existing nodes on the network and the nodes that distribute the transaction data, information distribution may be slow and temporarily cause “partiality in information volume.”

The PoR network structure resolves these problems that occur in a traditional decentralized system by making the system more structured.

With the PoR network structure, it is possible to regularly complete information distribution using a fixed number of processes without changing the information distribution process, even if the number of nodes were increased. This makes for a system with a high distribution speed that remains steady regardless of network size, in which information can be distributed almost simultaneously and “partiality in information volume” is unlikely to occur.

As described above, PoR makes use of the benefits of zero downtime and high security by having neighboring nodes monitor and supplement one another as a P2P decentralized network structure, while also incorporating some of the traits of an existing network structure in order to achieve the high-speed processing and “scalability” to successfully reconcile “speed” with “security.”

2. Scalability

2.1. The scalability problem

A major problem in existing blockchain technology is “scalability.”

The blockchain structure of the widely known Bitcoin system has a single-strand chain structure in which one block always connects to and extends from each block.

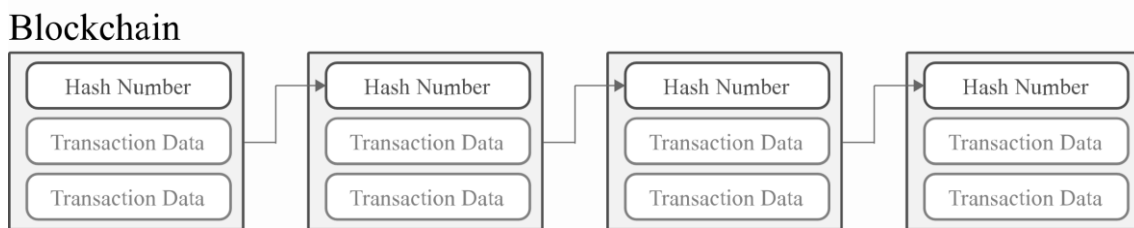


Fig. 2.1. Blockchain structure

Since this single-strand blockchain ensures consistency by appending all transaction data to the ledger in a batch at regular intervals, the consistency of all transaction ledger data and all transaction data must be accurately recorded. Accordingly, the approval process, i.e. mining, is required.

Blockchains can connect blocks by mining, but the nature of connecting blocks one by one causes an issue. There is a limit to the number of transactions that can be written for each block.

The size of one block in a Bitcoin blockchain is limited to 1 MB. Transactions that exceed this limit are placed in the next block, increasing transaction volume and causing a delay in data and a congestion in transactions.

A new technology called a DAG was created to solve this problem. DAGs tackled the scalability problem by improving the system in which blocks that group together the transactions that were in a blockchain are connected one by one.

2.2. DAGs

DAG stands for Directed Acyclic Graph, and it refers to a directed graph with no closed chains in graph theory. The DAG data structure has already been used in several projects worldwide (IOTA, Byteball, etc.), proving its useful and stable operation. Unlike blockchains, which approve one block containing a compilation of transactions, DAGs approve each transaction without creating blocks at all. This has allowed a major improvement in scalability by eliminating the need to create blocks and making it possible to approve transactions without the block size restrictions of a blockchain and without waiting for blocks to be created.

2.2.1. Existing DAG systems

In existing DAGs, transaction are approved by a system in which “the creator of the transaction approves past transactions.” In this system, repeated creation and approval by an unspecified number of nodes causes the chain to expand, guaranteeing the consistency of past transactions.

This DAG system has made it possible to solve problems such as block size in blockchains. However, it has not solved every problem. The problem of transaction approval time being dependent on the volume of transactions on the network and the transaction “finality” problem caused by the occurrence of forks still remain.

2.3. DAGs in PoR

PoR also uses DAGs. PoR uses an original DAG system that solves the problems present in existing DAGs and the problems associated with blockchains. The DAGs used in PoR are executed based on a system in which transactions are organized around timestamps generated by the signal nodes described below, and approval is given by master node groups that are assigned based on time axis.

First, transactions created by nodes are timestamped based on network time axis called “signals” and organized in order of creation.

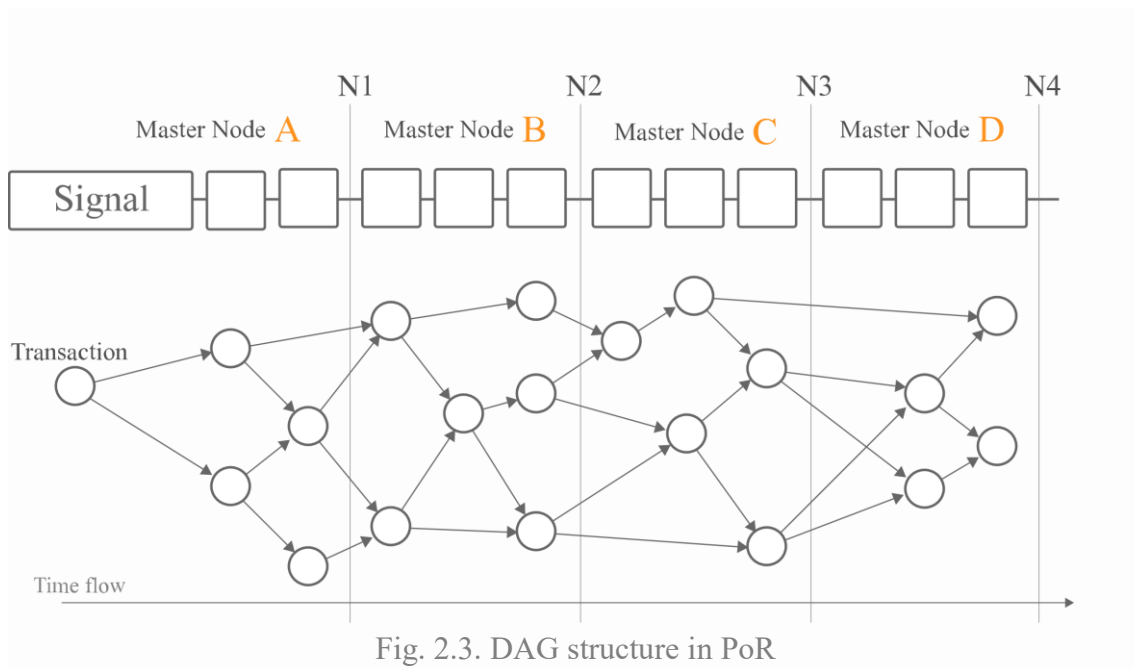


Fig. 2.3. DAG structure in PoR

Transactions organized by time axis are approved by the root node; at this time, the master node assigned as the root node for each time axis is determined using “signals.”

In the above FIG network, Master Node B is the root node and approves transactions created between time axes N1 and N2. In this way, PoR uses a DAG system with a structured approval format in which transactions are approved while the root node is switched between master nodes.

2.4. Benefits of PoR's DAG system

The structured DAG system used in PoR has the following benefits.

(1) Standardization of approval times

PoR transactions are approved by the root node. This solves a problem with existing DAGs in which there is a variance in approval times caused by lack of approval if there are no transactions. As a result, drastically shorten the time from transaction creation to approval.

(2) Elimination of inconsistent transactions

If a fault occurs in a transaction, that transaction will be destroyed without being approved. This eliminates situations in which legitimate transactions are not approved because subsequent transactions do not connect.

(3) The scalability problem

Even if transaction volumes increase due to an increase in nodes, the system responds by increasing the amount of MNs based on network status, which makes flexible scalability possible.

(4) The finality problem

With the PoR approval structure, the “forks” that occur in blockchains will not occur. This allows transactions with finality, unlike blockchains in which it was difficult to completely define a transaction.

2.5. Optimization of network structure

P2P intercommunication in a fully distributed network is conducted by information being exchanged between randomly connecting nodes, but this has the disadvantage of information distribution becoming more frequent the larger the network becomes.

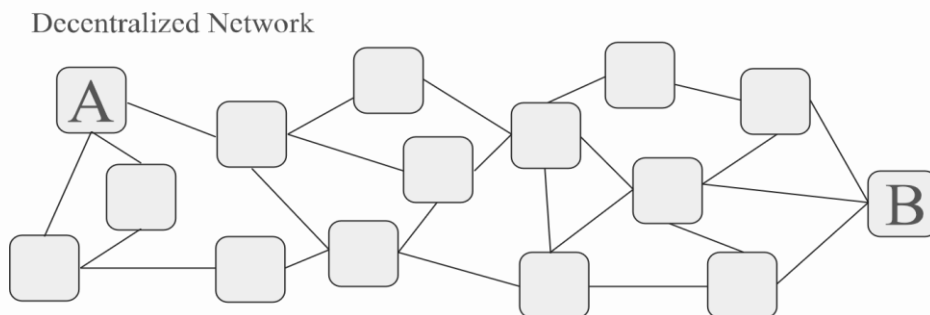


Fig. 2.5. Fully distributed network structure

For example, assuming a fully distributed network, information distribution from Node (A) to (B) in Fig. 2.2 is completed in a minimum of five times. However, in a massive network that contains many nodes, there will be an enormous number of routes between unevenly distributed node groups. This causes cases in which information does not reach the terminal if the information is not sent to a larger number of nodes. The fact that it takes time for information to be sent and received among these nodes leads to delays in the overall network, making it impossible for a standard P2P network to achieve high speed. PoR solves this problem by incorporating a structure known as a “node hierarchy.”

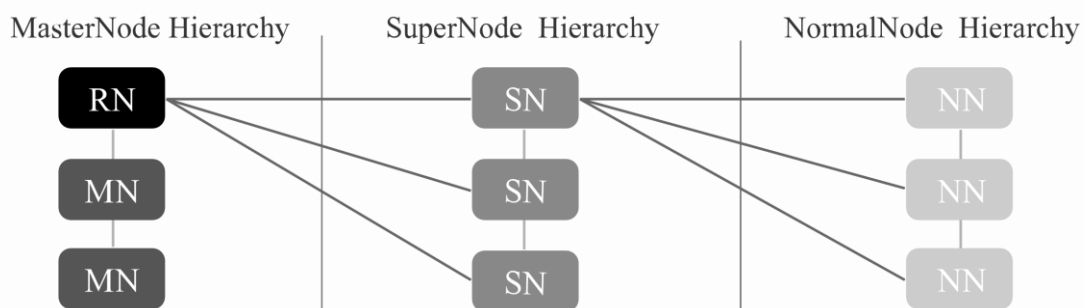


Fig. 2.5.1 Node hierarchy structure

Nodes composing the network were broadly divided into three ranks and connected together based on established rules, allowing previously irregular connections between nodes to be neatly organized. In network structures that use node hierarchies, it takes no more than two times for information from a terminal to be distributed to high-order MNs. Optimizing the connections between these nodes eliminates transaction delays that occur when sending information.

2.5.1 Round network

Along with its hierarchical network, PoR has a round network structure.

The previous section defined the problem of increasingly frequent communication between nodes when a network expands due to an increase in nodes, but PoR solves this problem with a round network.

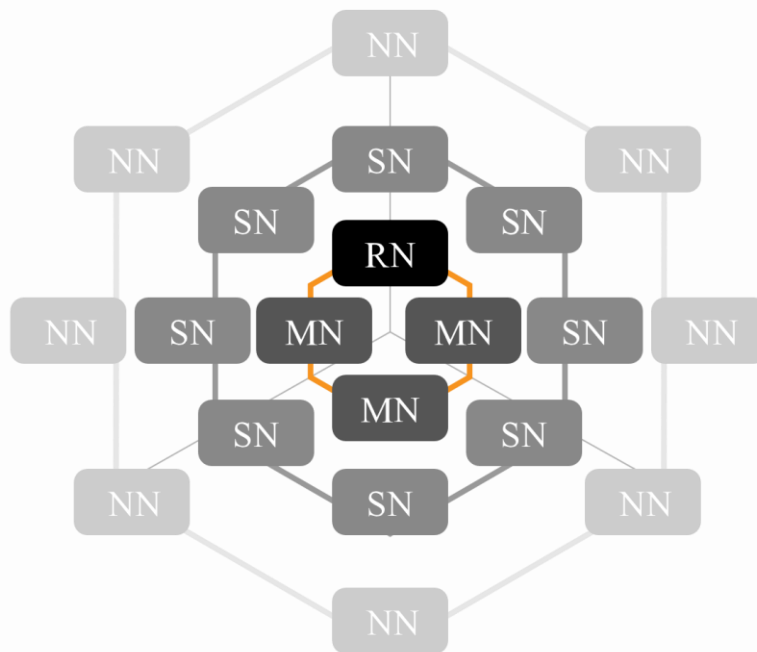


Fig. 2.5.1 PoR network

PoR's structure is simulated in Fig. 2.5.1. At the center is the master node group containing the root node. Around these are the super nodes. Finally, at the very edge are the normal nodes.

The number of nodes increase as the number of network participants increase, and even if the network structure expands, the organization from the center of the network to the terminal is always $MN \rightarrow SN \rightarrow NN$, a sequence which never changes.

In other words, the exchange of information from the center of the network to the terminal is fast and nearly equal regardless of network size. At the same time, this solves the “scalability problem,” which is a major problem in blockchains.

3.High Speed

3.1 PoR approval speed

One of the noteworthy benefits of PoR is “high speed.”

The following table compares the transaction processing speed (TPS) of PoR with other systems.

Table 3.1. PoR’s TPS compared to other algorithms

Exchange	Max transactions par second
Bitcoin	7 TPS
Ethereum	20 TPS
Pay pal (on Cyber Monday)	450 TPS
VISA	56,000 TPS
Proof of Round	40,000,000 TPS*

The TPS of PoR is roughly 6 million times faster than Bitcoin and 2 million times faster than Ethereum, which has the second highest aggregate market value. Furthermore, PoR is approximately 700 times faster than VISA, which is used as an actual payment service, a speed which surpasses all existing blockchain algorithms and is the fastest in the world.

Transaction approval speed is also faster than any existing blockchain at 0.2 seconds.





The detailed mechanisms of the high-speed consensus-building algorithm unique to PoR are explained below.

3.2. Assignment of roles to each node

The hierarchy is divided into three ranks in order to compose a round network. However, PoR also aims to further streamline processes by “assigning network roles” to each node.

These node roles are described below.

Table 3.2. Node roles and connections in PoR

	Connection node	Roles of each node
 RN Root Node	<ul style="list-style-type: none"> • Root Node • Master Node • Super Node • Nomal Node 	<ul style="list-style-type: none"> • Master node changes to the RootNode in a shift system. • Transaction consent. • Append to Blockchain.
 MN Master Node	<ul style="list-style-type: none"> • Root Node • Master Node • Super Node • Nomal Node 	<ul style="list-style-type: none"> • Distribution of transactions to SuperNodes. • Confirm Transaction consistency. • Monitoring the RootNode and other MasterNode.
 SN Super Node	<ul style="list-style-type: none"> • Root Node • Master Node • Super Node • Nomal Node 	<ul style="list-style-type: none"> • Confirm transaction consistency. • Distribution of transaction to each node. • Discard unnecessary transaction.
 NN Nomal Node	<ul style="list-style-type: none"> • Root Node • Master Node • Super Node • Nomal Node 	<ul style="list-style-type: none"> • Create transaction

NNs generate transactions and distribute them to SNs. At this time, SNs connect to a large number of NNs and receive the information. Then, SNs verify matches and distribute the information to MNs. MNs check consistency and distribute the information to the root node. The root node approves the transactions received and distributes the information to the MNs.

Giving each node an equal role eliminates anomalies that can commonly occur in the approval of blockchains and allows transactions to be processed smoothly.

3.3. Normal nodes (NN)

Normal nodes serve to create transactions and spread those transactions to the connecting normal nodes and super nodes.

In a hierarchical node structure, they are connected to one another based on the rule that “one node connects to five same-order nodes and five high-order nodes.” In other words, in a network structure, an NN is connected to five normal nodes including itself and five SNs and distributes transactions that it creates to all of these.

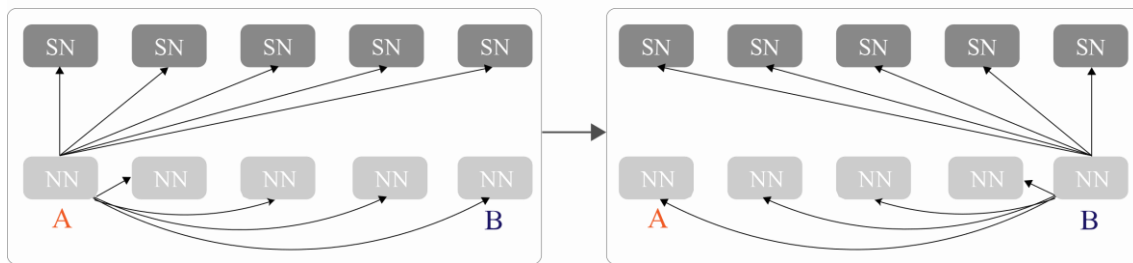


Fig. 3.3 Spread of a transaction by normal nodes

If NN (A) creates a transaction, that transaction is distributed to the NNs and five SNs. (Fig. 3.3, left)

NN (B), which receives the transaction next, distributes the transaction to the five SNs connected to itself. (Fig. 3.3, right)

In an actual network, NNs are connected to SNs randomly, so repeating this distribution process allows the transaction to be spread to all NNs and SNs.

By this logic, each node would receive the same transaction multiple times; however, the nodes process duplicated transactions by using “verification.”

3.4 Super nodes (SN)

Super nodes serve to validate transactions received from normal nodes and distribute them to master nodes. In PoR's network structure, SNs connect to multiple NNs, receive transactions from them, and verify their consistency.

However, as described above, NNs distribute transactions received from other NNs to the SNs connected to themselves, so an SN receives the same transaction from each NN connected to it.

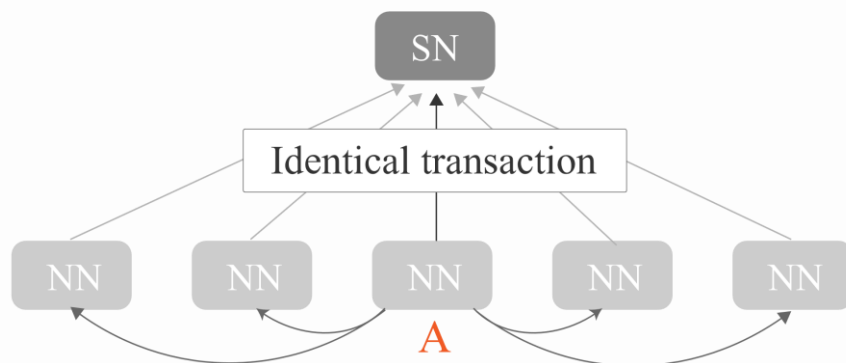


Fig. 3.4. Distribution of transactions from normal nodes to a super node

In the above figure, five NNs distribute a total of five “transactions with the same content” to one SN.

For example, imagine a PoR network structure in which as many as 1,000 NNs are connected to one SN. Applying simple arithmetic, when an NN creates and sends one transaction, 1,000 instances of the same transaction are distributed to one SN.

In order to ensure that only the necessary transaction among these is sent to the MNs, the SN “verifies” the received transactions before distributing them to the MNs.

3.5. Verification of transactions by nodes

After receiving a transaction from another node, each node verifies the transaction to determine whether or not it is already present. Based on the results of this verification, one of two processes is completed.

(1) If the transaction is not present

→The transaction data is saved to storage, its consistency is verified, and it is distributed to each node.

(2) If the transaction is present

→The transaction is destroyed.

The above logic is applied commonly to all nodes.

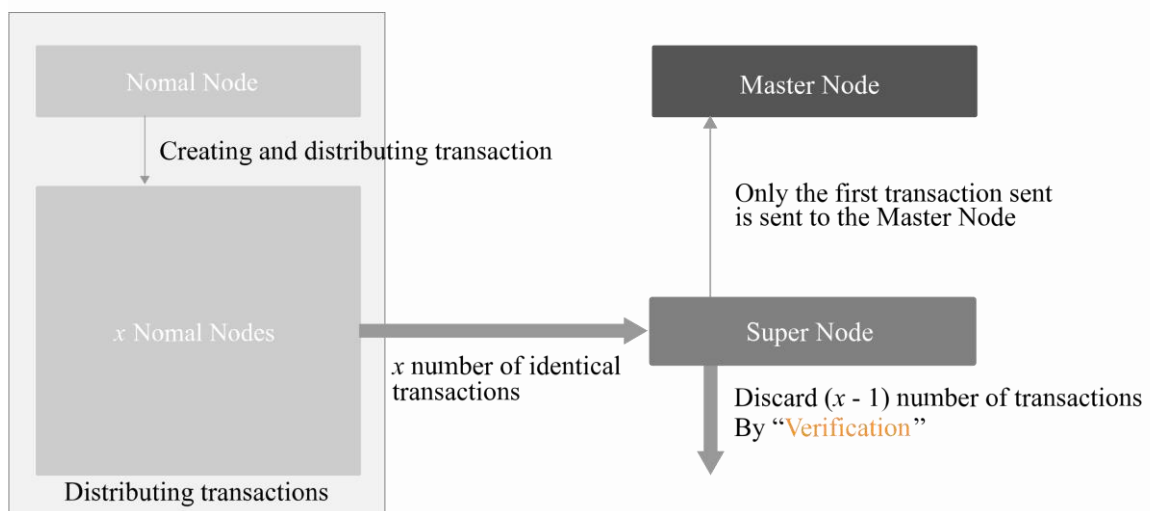


Fig. 3.5 Destruction of transactions through verification

Under the sample network conditions (Fig. 3.5), when one transaction is distributed to x NNs, the number of instances of the “same transaction” distributed to one SN is x . Furthermore, according to the rules of the network structure, an NN is connected to five SNs, so simple arithmetic shows that a total of $(5x)$ instances of the “same transaction” will be sent to the high-order nodes.

If these transactions were sent to the MNs as they were, processing this enormous transaction volume would place a heavy load on the MNs, causing a delay in their operation. In order to reduce the load on MNs, SNs are placed between NNs and MNs so that they can destroy unnecessary transactions through verification, thus serving to distribute the minimum number of transactions to the MNs.

3.6. Master nodes (MN)

Master nodes serve to verify the consistency of transactions that have been verified and distributed by super nodes and distribute them to other MNs and the root node.

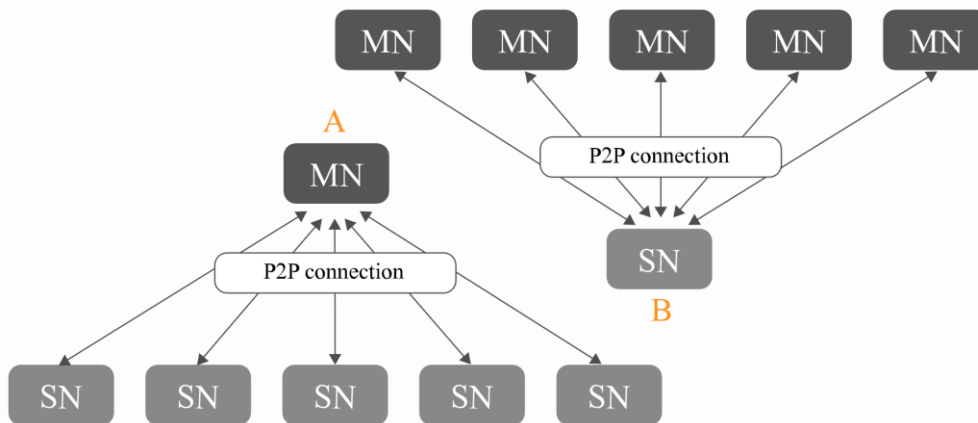


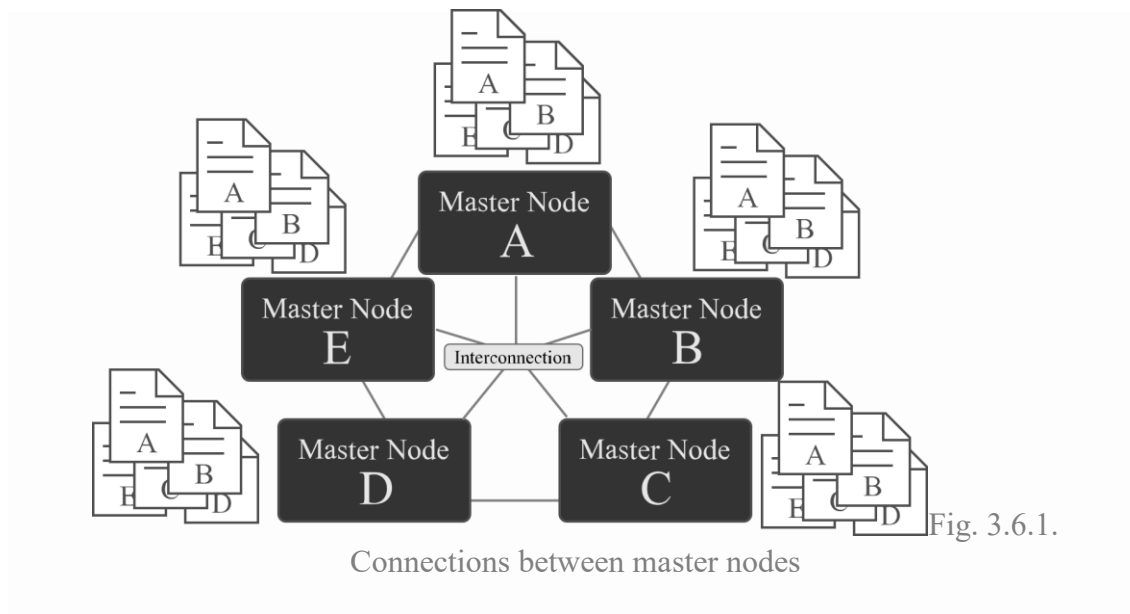
Fig. 3.6. Super node connections to master nodes

First, SNs are connected based on the rule that “a node is connected to five same-order nodes and five high-order nodes,” in the same way as the relationship with the NNs above (Fig. 2.7.1). MNs are also interconnected with five SNs (Fig. 2.7.2).

Furthermore, MN (A) receives the same transaction from five SNs, but it verifies the transactions in the same way as the SNs and destroys transactions that are already present.

3.6.1 Connections between master nodes

Unlike other nodes, master nodes are structured based on the rule that “all active MNs are interconnected with all active MNs,” as shown in Fig. 3.6.1 below.



After receiving a transaction from a connecting SN, MN (A) “verifies” that transaction and determines whether or not it is already present. If it is present, it destroys the transaction; if it is not present, it verifies its consistency. A transaction whose consistency has been verified is quickly and simultaneously written to the MN’s own HDD and distributed to all other connecting MNs (B) through (E).

After receiving the transaction, MNs (B) through (E) check once again whether the transaction is already present and verify its consistency.

Since all MNs are interconnected, it is possible to efficiently and quickly spread transactions gathered from the entire network to the MNs.

3.6.2. Appending to the round header

There is an important chain contained in PoR called a “round header.” This round header does not contain information on PoR transactions; instead, MN group information is appended.

The information appended is as follows:

- Master nodes that participate in the current network
- Voting results of yes or no for requests to add a new master node
- Block creation time

Appending to the round header does not depend on time setting; instead, appending is conducted by the MNs when there is a change in MNs contained in the network, such as an MN stopping or starting, a new master node participating, etc.

In order for a new master node to participate, consent must be obtained in a majority vote by the master nodes currently operating on the network. By publishing the results of this vote on the round header, MNs participating and operating in the network can refer to the round header to learn that the new MN has obtained the trust of other master nodes.

3.7. Root nodes (RN)

The root node serves to approve transactions.

In the PoR system, the RN exists with the same rank as the MNs. However, the role of RN is not assigned to one node all of the time; approval is given by the MNs that share the role of RN in sequential order in accordance with the rules.

In order for transactions to be approved by the RN, the MN assigned to approve transactions within a certain time frame is determined in accordance with the rules based on the PoR network’s system time, and approval is given by “that assigned MN” at “that assigned time.”

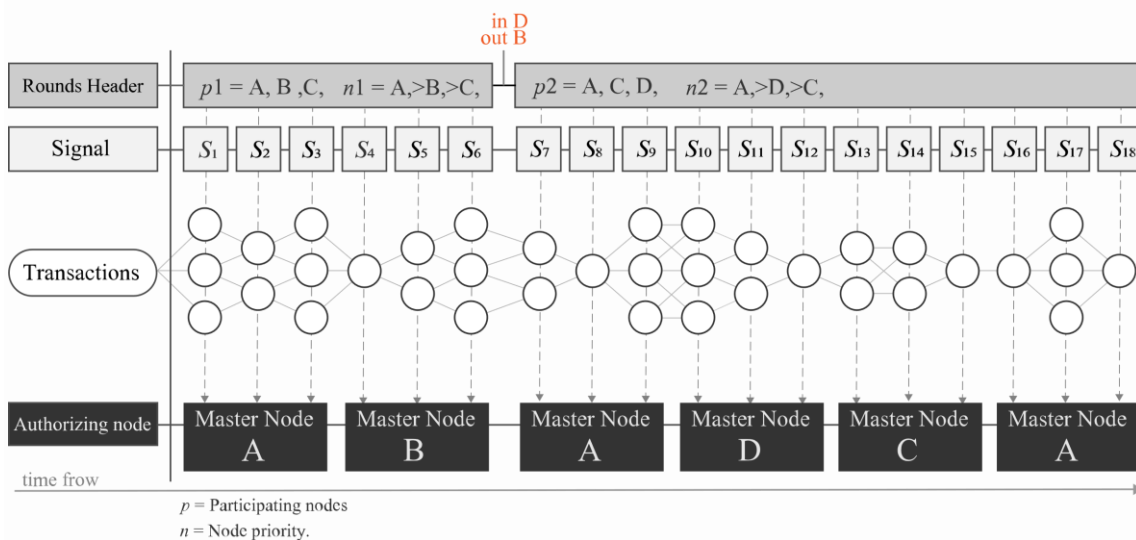


Fig. 3.7. Switching of the root node based on time axis

Distributed transactions have a system time hash value. Based on this hash value, transactions with a hash value between time axis S_1 and S_3 are approved by MN.A. Those with hash values that pass time axis n_3 are approved by MN.B.

In this way, the MN that is assigned in advance based on the time axis becomes the RN, approves the assigned transactions, and connects the DAG.

MNs that are not assigned based on a time axis cannot give approval; instead, they supplement and monitor the RN by verifying consistency.

3.8. Synchronization between nodes using signals

PoR uses timestamps that use a logical clock in order to synchronize each node group within PoR.

In the “distributed computing” field, it is not easy to obtain a consensus regarding the time taken for each node. With a centralized network, it is possible to determine the sequential relationship based on blocks in which the time axis is shared globally. However, in a dispersed system, clock value errors in various nodes existing on the network and communication-based time lags make it extremely difficult for time to be shared completely.

PoR conducts synchronization within a network by placing multiple “signal nodes” within the network that create and distribute timestamps called “signals.” Signal nodes are placed within normal node groups and serve to create signals and distribute them to each node just like transactions.

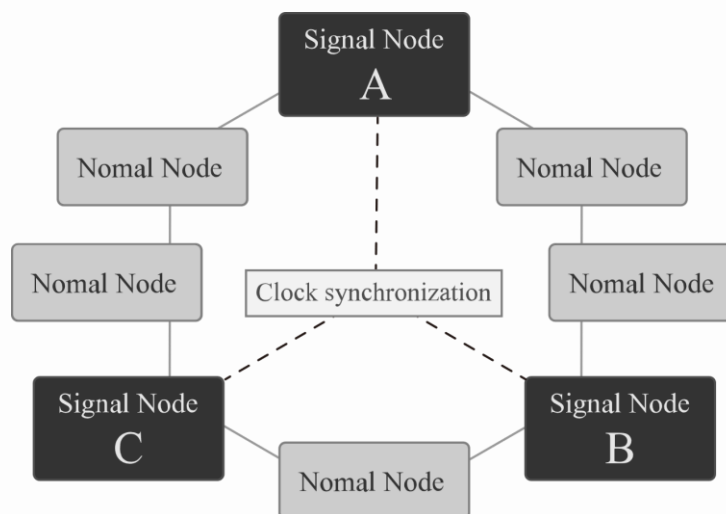


Fig. 3.8 Placement of signal nodes on the network

Signal nodes (A) through (C), which exist evenly throughout the network, each exist as nodes, connect to NNs, and distribute timestamps; when an error occurs in the clock value between these timestamp nodes, network synchronization does not operate properly.

Therefore, in PoR, rather than having timestamp nodes operate independently, they are always interconnected from behind, and network

synchronization is conducted by synchronizing clock values with one another in order to eliminates signal errors.

3.9. Elimination of nonces

In the approval process of blockchains like Bitcoin, the “nonce” system is used to verify consistency. Through the calculation of a large amount of nonces by many miners (mining), blocks are created and the consistency of bitcoins is guaranteed. However, appending bitcoin blocks in order to mine requires a significant amount of time and an enormous amount of electricity at a major cost to the system and indirectly to the environment.

By eliminating the calculation of nonces, PoR has successfully reduced the time it takes to mine and greatly shortened transaction approval time.

In PoR, transactions are approved by the root node, but the role of root node is not always assigned to one node; instead, the role is switched to a new MN every second.

When this occurs, the decision regarding which MN will be chosen as the next root node is made based on the following rules established in advance which determine the sequence:

- Node age
- Number of transactions successfully approved

Since the above factors refer to past PoR data, this information allows the permutations to be learned by any node, maintaining impartiality in the network.

The following figure describes the process in which the master nodes on the network switch out the role of root node every second and approve transactions.

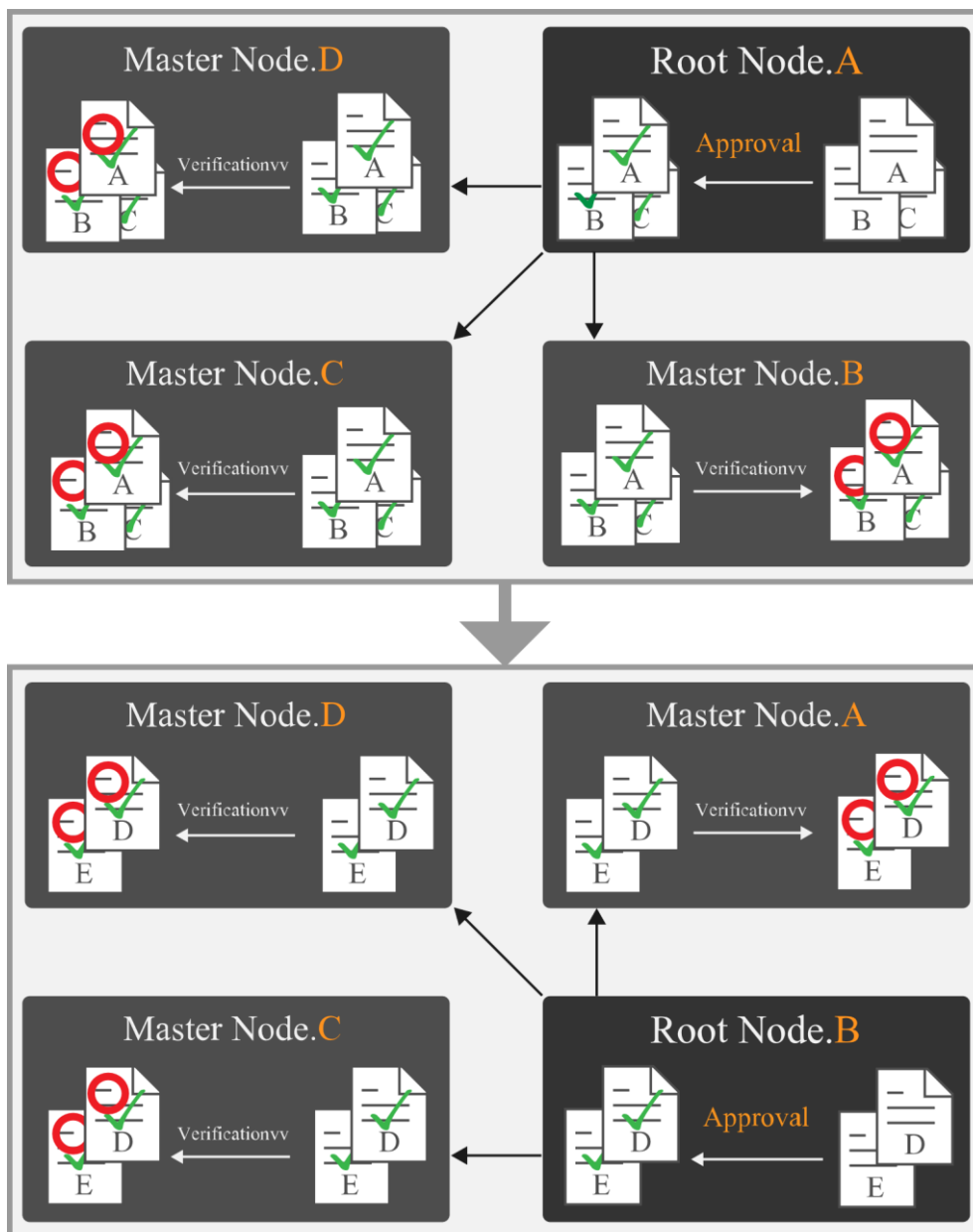


Fig. 3.9. Switching the route node

First, MNs participating in the network are gathered and their sequence is determined by sequence rules. In this way, they are placed as shown in the figure and (A) becomes the first root node. The root node approves transactions received from the MNs and returns them to the MNs. When this occurs, the MNs monitor one another, verify consistency, and distribute the information to the SNs. One second later, (A)'s turn as root node ends, (B) becomes the next root node, and verification begins.

In this way, by determining in advance the sequence of root nodes that will approve transactions, it is possible to shorten the time it takes to select the root node and make the switchover smooth allowing resources to be devoted to the approval process.

In addition, constantly switching the root node protects the network by distributing cyber attacks and heavy loads that can occur on a centralized system.

Furthermore, in an actual network structure, multiple root nodes exist at the same time relative to the size of the network, and they approve transactions concurrently.

For this reason, the capacity for processing transactions in PoR as a whole is determined by the total arithmetic capacity of the nodes participating in the network; the more master nodes increase, the more transaction processing speed increases.

In addition, eliminating nonces makes it possible to completely eliminate the mining required by existing algorithms, which in turn greatly reduce the amount of energy needed to maintain the network.

3.10. Speed summary

- Optimization of network structure by using a round network
- Optimization of processes by assigning roles to each node
- Reduction of unnecessary transactions using super nodes
- Approval of transactions without nonces
- Synchronization of nodes using timestamps
- Improvement of TPS through the efficient use of node specs

Due to the above factors, PoR has been able to successfully build a scalable network with an approval speed of **40 million TPS** that beats the record set by existing algorithms.

4.High Security

4.1. Security

The ability to offer high security is one reason that blockchains have received so much attention globally. In conventional centralized systems, information is gathered in one place. This makes it easy for attackers to specify an attack target, and there is limited ways to manage the rewriting of information when the event of a successful attack. System administrators are constantly pouring significant time and money into robust security to prevent attacks from being successful.

The newly developed blockchain technology and dispersed ledger approaches were methods that would supersede existing systems by intentionally dispersing information among network participants so that it is managed and protected by all of them.

Rewriting information appended to a blockchain requires rewriting the ledgers held by network participants, the calculations necessary for which are immense, making unintentional manipulation of a blockchain practically impossible.

When looking at the differences between a centralized system and a blockchain in this way, many people tend to conclude high speed and high security to have a trade-off relationship. However, PoR has successfully maintained high security whilst achieving record-breaking high speed. This is possible as it sacrificed the factor of “impartiality” by incorporating a system known as a “node hierarchy” into the blockchain infrastructure.

Centralized systems are vulnerable in that they represent one attack target and if an attack was to succeed, the system’s information could be manipulated easily; however, protecting this with dispersed master nodes decentralizes attack targets like blockchains and makes the system resistant to rewriting. In addition, nodes repeatedly check consistency using encryption key technology, which thoroughly eliminates malicious hacking and ensures transaction consistency.

By sacrificing the “impartiality” of having only some network participants approve information, PoR has successfully achieved high security.

4.2. Hash functions

A hash function is a function that creates a bitstream of a fixed length and no fixed form out of a bitstream of arbitrary length.



Fig. 4.2. Conversion by hash function

Bitcoin uses a hash function called “SHA-256.” This function turns data into hash values that are 256 bits long, and it has an extremely strong resistance to manipulation. It is also frequently used in other IT fields, which proves its usefulness and reliability.

PoR uses “SHA-512,” which creates doubly long hash values of 512 bit length. This is considered the safest cryptographically among the “SHA-2” hash function standards established by NIST (National Institute of Standards and Technology), which also include “SHA-256.”

Hash function “SHA-512” has the following characteristics:

(1) Irreversibility

Irreversibility is the inability to return a converted hash value to its original form using a function. This makes it impossible to revert, view the underlying data or manipulate the transaction content that has been converted to a hash value.

(2) Collision resistance

One problem that hash functions face is collision of hash values. This refers to cases in which, for example, given an input m_1 , $\text{hash}(m_1) = \text{hash}(m_2)$. For hash values for which a hash value collision is likely to occur, it is possible to switch legitimate text with illegitimate text through a “preimage attack.”

PoR converts extremely important and large-volume transactions to hash values. It heightens system security by using “SHA-512,” which has a high collision resistance, so that calculated hash values cannot be manipulated by “preimage attacks.”

4.3. Digital signatures

PoR ensures transaction consistency by using digital signatures.

A digital signature is a technology that verifies that there is no error in the destination of data received and that data has not been manipulated during the transmission route. This technology is guaranteed by a pair of encryption key technologies called public keys and private keys:

(A) Public key

A key used to encrypt data. Data encrypted with a public key can only be reverted with a private key.

(B) Private key

A key used to decrypt data. Data encrypted with a private key can only be reverted with a public key.

As a general rule, each node has a pair of keys containing one encryption key and one public key, and each node stores its encryption key so that only that node can locate it.

The process of verifying transaction consistency is as follows.

(1) Conversion using a hash function

First, the NN converts the transaction data to a hash value using the “SHA-512” hash function.



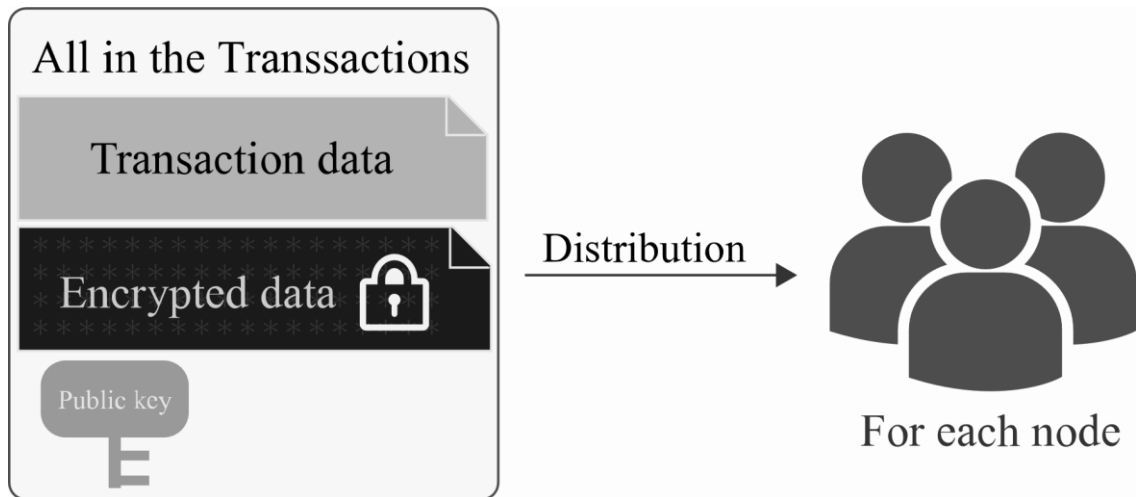
(2) Encryption

It then encrypts the calculated transaction hash value with its own private key.



(3) Transaction distribution

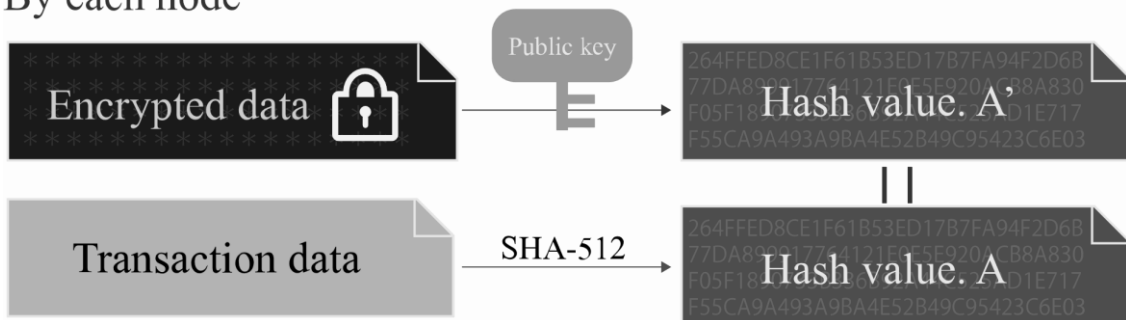
The encrypted data and public key are packaged with the transaction data that has been converted to a hash value, made into a transaction, and distributed to each node.



(4) Consistency check by digital signature

The node that receives the transaction calculates the hash value of the transaction data using a hash function, “reverts” the concurrently attached encrypted data using the enclosed public key, and checks its value.

By each node



Data encrypted with an encryption key can only be reverted with a public key. For this reason, the hash value of the reverted transaction data will match the hash value of the enclosed transaction data as long as there has been no malicious rewriting. If it does not match, it is determined that “rewriting of the transaction has occurred,” and the SN or MN will destroy the transaction.

Verifying transactions with a digital signature guarantees consistency in the creation, distribution, and approval of transactions.

4.4. Elliptic curve cryptography

To create public keys from private keys, PoR uses an elliptic curve cryptography technology called “Secp256k1.”

“Secp256k1” is the encryption key technology used by Bitcoin, and its robustness has been proven by Bitcoin, which has operated without any system hacks since 2009.

The algorithm process is as follows.

•Creation of the public key

(1) The formula for an elliptic curve in elliptic curve cryptography is as follows.

$$c^2 = x^3 + ax + b \text{ mod } p$$

(2) First find a, p, b in the above formula, then find the point of reference $G(x, y)$. Secp256k1 uses the following parameters.

a =

0x00

b =

0x0007

p = 0xfffeffffc2f

Gx = 0x79be667ef9dcbbac55a06295ce870b07029bfcdb2dce28d959f2815b16f81798

Gy = 0x483ada7726a3c4655da4fbfc0e1108a8fd17b448a68554199c47d08ffb10d4b8

(3) Add the node’s $G(x, y)$ to this $G(x, y)$ and repeat that addition function n times (n =value of the node’s private key). This will give you the value nG .

(4) Use the obtained value as a public key.

It is easy to find nG from G by repeating the modulo operation n times, but when n is an extremely large value, it is considered extremely difficult to find n from G and nG using the modulo operation. This creates unilateralism in key creation just as with a hash function, making it impossible to decipher the encryption. This extremely robust encryption key system guarantees the security of private keys.

4.5. Rigorous consistency checks by each node

Section 4.3 explained that transaction consistency is guaranteed using digital signatures, but this section will explain how rigorously transaction consistency is verified on the network.

First, transactions created by normal nodes are distributed to the root node and approved. At this point, they pass the super nodes and master nodes that exist between the root node. Furthermore, based on network rules, the transaction is also distributed to multiple high-order nodes and same-order nodes. In PoR, all nodes to which a transaction is distributed are set to verify the transaction separately and eliminate malicious hacking.

At each node, if a received transaction is not already present, its consistency is verified, and those deemed valid by a consistency check are distributed to a higher-order node.

Of course, it is also conceivable that a transaction sent by a malicious node may be fraudulently rewritten before being distributed to each node.

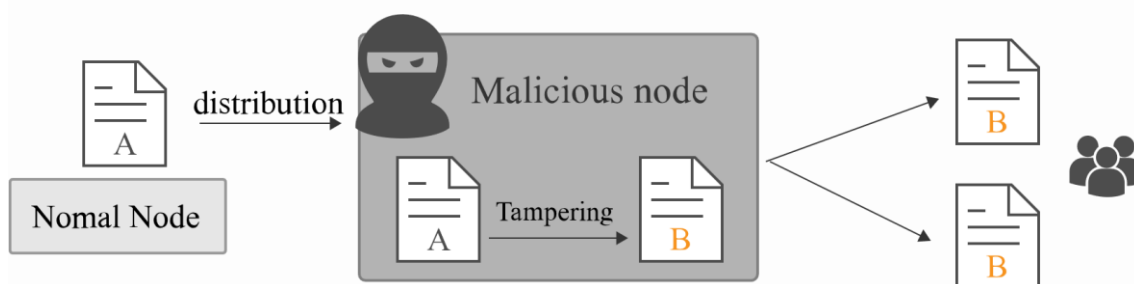


Fig. 4.5.1. Transaction rewriting by a malicious node

A transaction that has been distributed after being fraudulently rewritten will be received by the destination node. However, the transaction's consistency will be checked using a digital signature by the destination node.

Transactions found to be manipulated by this consistency check will be deemed fraudulent and will be destroyed by the destination node.

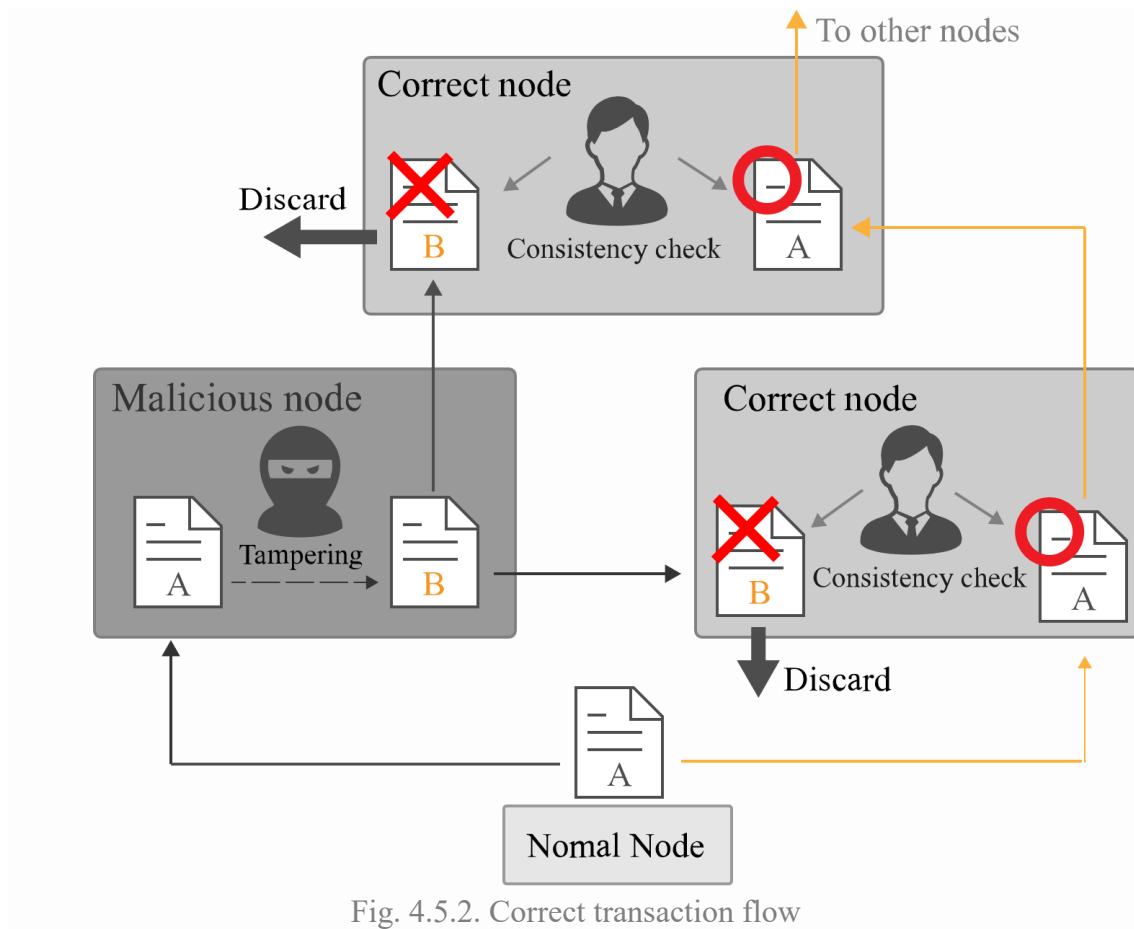


Fig. 4.5.2. Correct transaction flow

While transactions destroyed by the node will be erased, legitimate transactions will be distributed properly through other nodes (green lines).

In the PoR network, nodes check the consistency of each transaction received, which thoroughly eliminates malicious transactions; at the same time, this optimizes the structure so that legitimate transactions are distributed smoothly.

4.6. Unspecifiability of signal nodes

PoR network synchronization uses signals to create signal nodes. However, if a signal node is specified and maliciously attacked, there is a risk of a discrepancy occurring in network synchronization that could become a vulnerability.

In order to compensate for this vulnerability, signal nodes are disguised as NNs and camouflaged to make it difficult to specify the nodes from outside.

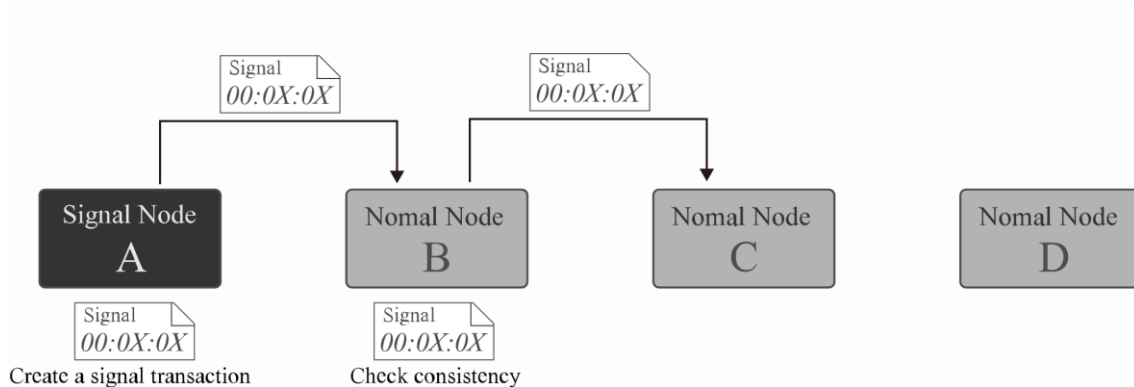


Fig. 4.6. Signal transmission

For example, Signal Node (A) creates a signal and distributes it to NN (B). After receiving this signal transaction, NN (B) conducts verification and then distributes the signal transaction to Node (C) connected to itself. From the perspective of NN (B), (A) distributed the signal, so it can presume that (A) is the signal node. However, from the perspective of NN (C), NN (B) distributed the timestamp, so it can presume that (B) is the timestamp node. When this distributed transaction is sent to other nodes, due to the system characteristics, all NNs act like signal nodes from the perspective of each node, making it difficult for the true signal node to be specified.

4.7. Management of signal nodes

Signal nodes themselves are managed so fraudulent access could be denied even if a signal node happened to be specified or successfully attacked.

Furthermore, since multiple timestamp nodes exist on the network, even if one node were to go down, the synchronization of clock values on the overall network would be executed by other regularly operating timestamp nodes.

Even if a malicious node were to rewrite and distribute a timestamp, the transaction would be deemed fraudulent by a digital signature consistency check and destroyed.

4.8. Reconstruction of network structure

If a change occurs to the network due to participation of a new node, withdrawal of a node, etc., each node in the PoR network will detect it and network reconstruction will occur. Since reconstruction is conducted separately for each node every 30 seconds, connections between new nodes are continuous on the PoR network. In addition to improving network recovery, this also makes it difficult to attack a specific node by having a network that is constantly reconstructed and continues to operate and it also makes it possible to disperse the load on the network.

4.9. Resistance to attacks on nodes

With existing centralized systems, many servers are still exposed to DDoS attacks. Decentralized blockchains have a certain amount of resistance to these attacks due to the dispersion of information. The PoR network is a dispersed network incorporates the following three measures against these attacks:

- Anonymous IP addresses
- Reconstruction of network structure
- Unidentifiability through dynamic IP addresses

(1) Anonymity through IP addresses

Managing accounts and identifying nodes based on public keys means that IP addresses are not associated with public keys, which serves as pseudo-concealment for IP addresses and makes it difficult to specify an attack target.

(2) Reconstruction of network structure

Nodes update their connections every 30 seconds, so if a node goes down the connection nodes will rapidly connect to new nodes. If an MN goes down, an SN temporarily takes over the role of the MN at the time of network reconstruction in order to fill the hole in the network.

(3) Unidentifiability through dynamic IP addresses

Managing nodes using AWS, GCP, etc. means that even if a node goes down it will be rebooted automatically. Furthermore, the use of dynamic IPs means that the IP address will change at each reboot, which makes it impossible to specify an attack target and therefore difficult to attack.

4.10. Elimination of malicious hacking based on economic rationality of nodes

In the PoR network structure, the role of an SN is to verify the consistency of network construction transactions and distribute the transactions to each node.

In addition to verifying consistency, MNs monitor one another, approve and distribute transactions as root nodes, and play the extremely important role of acting as the center of network construction.

If specifications allowed anyone to easily build nodes that could serve as the center of the network, an influx of malicious nodes could clearly occur. PoR controls the influx of malicious nodes by setting strict and rigorous conditions for building these nodes, and malicious hacking is thoroughly eliminated by establishing restrictions that prevent malicious nodes from being approved.

For example, one system that uses the PoR is “BEXAM” BEXAM is a dispersed platform that uses the PoR system as a basis for its services.

In order to build SNs and MNs on BEXAM’s network, it is necessary to purchase a certain number of “BXA tokens” within the BEXAM and “lock them up”. The lockup process requires trading fiats and tokens, which are by no means cheap. In this way, establishing restrictions on the construction of nodes that make up the center of the network raises the hurdle for node construction and prevents an influx of malicious nodes.

However, even with the hurdle raised, it is impossible to completely eliminate the possibility of the construction of malicious nodes. For example, a transaction could be rewritten by a malicious MN, or a root node could give malicious approval. While the transaction itself will be destroyed by another MN, the malicious MN itself cannot be neglected.

In this case, BEXAM takes the approach of freezing the MN. This also freezes tokens used in the lockup process, essentially erasing the funds. By holding locked up tokens hostage, BEXAM uses economic rationality to eliminate malicious hacking.

4.11. Security summary

- Use of the “SHA-512” hash function
- Use of digital signatures
- Rigorous consistency checks by each node
- Dispersion and unspecifiability of timestamps
- Recovery through dispersion of attacks on nodes and reconstruction of network structure
- Elimination of malicious hacking based on rationality

Due to these characteristics, PoR has created a structured dispersed system whilst allowing high speed/scalability and high security.

Specifications

The document is Version 0.2 of the Proof-of-Round white paper.

The latest information on new technology will be distributed through this document.

This document will be updated in accordance with future technological advances. For the latest information, please refer to the Proof-of-Rounds white paper, the community, developers, and the EARLY WORKS CO., LTD. website.

Copyright declaration

The copyright for this document belongs to EARLY WORKS CO., LTD.

Disclaimer

EARLY WORKS CO., LTD. will revise the technical white paper as needed based on technological updates of Proof-of-Rounds.