NOSO

Network Of Sustainable Opportunity

Protocol 2

WHITEPAPER

Authors: Noso Community

Updated: May 2023, Version 5

WHAT IS NOSO?

Noso is a dynamic and sustainable Peer-to-Peer cryptocurrency that is accessible to consumers and merchants, is fast, reliable, and secure with minimal transaction costs. Noso has its own lightweight blockchain which is not a fork, token, or copy of any other existing codebase. With Noso's consensus ensuring decentralized masternodes (MN), the network is immune to 51% PoW attacks. The project goal is to create a unique and sustainable cryptocurrency with equitable distribution that will be widely used for the payment of goods and services online, without a need for third parties or intermediaries.

NOSO PROJECT OBJECTIVES AND FEATURES

The Noso project vision was not to just create another standard cryptocurrency. The creator's idea was to build a fast and secure currency based on the philosophy of its predecessors, while addressing flaws and adding functionality no other cryptocurrencies has.

Blockchain growth size over time is a common concern for many cryptocurrencies. As a coin's blockchain ledger grows, its network speed and efficiency slows down. Noso addresses these issues by removing the need to consult the blockchain for authenticity of its status. This is done by keeping its merkel tree and address balances separate. In the Noso protocol, blocks serve to order transactions correlatively and keep the account summary updated. Once a block is created, its information is added to the blockchain. From that moment forward, the newly created block's utility lies only in consulting the transaction history. For this reason, Noso is independent of the blockchain once a user updates their block to the network. These design elements not only help solve blockchain size scalability issues and increase transaction speed, they also allow for efficient enhancements and updates to the Noso protocol.

Built into the Noso protocol is the **concept of assigning an "alias"** to your public address, referred to as "customizable address transfer." This customization consists of assigning an alias "name" to a specific Noso address so that it is easier to remember. Instead of a user sharing their 30-31 character hashed public address for receiving funds, they can share their easy-to-remember "alias" address. (example: Noso Address: *N2kFAtGWLb57Qz91sexZSAnYwA3T7Cy,* Address Alias: *devteam_donations*) An address that has been customized is charged a nominal fee (0.00025000 Noso) and is immediately registered to the summary of accounts. Alias name changes can only occur one time per Noso address and cannot be duplicated on different addresses. Noso "aliases" help to make Noso easier for the community to use.

In addition to aliases, the Noso creator's vision was to create a way for the community to easily monetize their Noso currency without a need for third party exchanges. **Nobiex** (Noso built in exchange) is under development and will allow exchange of Noso into other cryptocurrencies using the Noso blockchain as escrow. Once fully deployed, Nobiex will allow the Noso community to participate in decentralized peer-to-peer Noso monetization using their Noso Wallet.

The creator also envisioned a **"parallel" coin** which runs in tandem on the same Noso mainnet. This coin, currently referred to as Kreditz (Final name will be approved through GVT vote process), is a parallel coin reserved for future use to be announced. In addition to this future-use coin, there are also Governance Tokens (GVT), used for the Noso poll management/voting system (read more below under the Noso voting system section).

Noso is a Proof of Stake **"Masternode coin"** and provides investors the opportunity to earn guaranteed income from operating a node. Nodes not only offer a steady form of income, they promote decentralization and network stability on the Noso network. Masternode investors are required to maintain a Noso balance of 10,500 coins and a system running NosoNode software on a 24-hour basis. Masternode rewards are earned upon each block completion and can either be sold or held toward investment in another masternode. NosoNode software is open source and is publicly available for download on the Internet. (See nosocoin.com/docs for more details on operating a Masternode. See the Noso Cryptonomics section for details on MN rewards).

In order to earn, hold, send, or receive Noso coins, a wallet is required. The Noso Project provides two wallet app options; **NosoLite** and **NosoMobile**. Both wallets allow users the ability to generate new addresses, assign aliases to addresses, import or export addresses, password encrypt addresses, and send and receive coins to and from other Noso coin holders. NosoLite is available for most major operating systems including Windows, Linux and macOS. NosoMobile is available for Android. Once installed, both wallets will auto-create one default address and are ready for use (See nosocoin.com/docs for more details on these apps).

To keep the status of the Noso network up-to-date at all times, **blockchain explorers** have been freely created by the community to serve as the source of information for the protocol. These explorers show information pertinent to the protocol such as block height, supply, orders, rewards, fees, coin distribution, and Node information among other things. (See nosocoin.com/docs for more details on explorers).

IMPROVEMENTS OVER PREDECESSORS

The Noso Project recognizes contributions made by its predecessor, Bitcoin, and strives to address its flaws while adding functionality no other cryptocurrencies have. With this in mind, we will contrast Bitcoin to many of the features built into Noso.

Block Similarities

Noso, like Bitcoin, has its own chain, which includes all blocks created from block 0 to date. The first block in both BTC and Noso is referred to as the 'Genesis block'.

Every BTC block has a transaction at the beginning called “Coinbase”. In this special transaction, the quantity of new coins created in that block is mentioned. The quantity is given and follows the programming guidelines set forth by the creator of the coin. In Noso, there is no “Coinbase'' transaction in the block “body,” rather coin quantities are mentioned in every block “header.” Regardless of this block location difference, in Noso a programming command is what allows each block creator the ability to mint a designated number of new coins. (This process would be akin to placing an order to a coin mint factory and asking them to create a certain number of new coins). This function is responsible for the base production of new coins and is referred to as 'Coinbase.’ Like BTC, the creator of each block always receives the entire amount of newly minted coins, however, in the case of Noso, this amount is divided into 3 categories (Noso MasterNodes, Project development funds, and Proof of Participation on Work).

Transactions/orders

Much like Bitcoin has “transactions,” Noso has”orders.” In a BTC, “transaction” shipments can be made up of many addresses and many recipients. In contrast, Noso “orders” can have many senders but only one receiver.

In BTC, having multiple recipients is imperative because of its philosophy of unspent previous transactions (UTXO). Typically transaction amounts are never equal so it must also be possible to return excess amounts from transactions to the sender.

In Noso this problem does not exist because actual balances of all existing accounts are kept in separate ledgers and are updated after every block. A Noso transaction does not look to find how many unspent transactions a sender has, but rather looks at the ledger of the balances and verifies funds are available to send. If a sender’s account balance is great enough to fulfill the order, it is transmitted on the next block; the ledger is then informed of this movement and adjusts sending and receiving accounts accordingly on the following block. (See Noso Transactional Operation section for more detailed information on transactions).

Block Creation

In BTC, the system recalculates the difficulty level of the algorithm every 2,016 blocks, allowing whomever found a solution to create the block. In Noso the difficulty is almost

infinite in all blocks; Block creators are those who, before the current block time expires, will have found and announced to the network the best possible approach to the target. In this way, instead of block time being variable (BTC), in Noso it remains absolutely stable (10 minutes).

Merkle root

In BTC the field of 'merkle Root' is embedded in the block. In Noso this is kept in a separate ledger. This "general ledger" keeps balances of all accounts containing a non-zero balance. This general ledger is recalculated every 100 blocks and addresses concerns with both security and calculation speed as well as easier resolution when the network presents a problem.

These pioneering methods of information management, in contrast to its predecessor, allow Noso to operate safely with only the last block and its general ledgers. This design is not only pioneering, it vastly reduces wallet build time and greatly improves the end user experience.

NOSO CRYPTONOMICS

Max supply: 21 million total supply
Genesys block: March 7th, 2021
Block reward: Starting with 50 Noso and halving every 210,000 blocks (~4 years) up to 10 times (~40 years). Block reward is split between Masternodes 90% and Project development funds 10% (beginning at block 88406).
Block time: 600 seconds
Transaction per second: Over 1900/second on final version with high specification masternodes hardware, currently mainnet average is ~50+/second.
Transaction Verification: <3 seconds
Difficulty: 16 exp 32 all blocks.
Ticker: NOSO
Noso has no ICO. A premine was deemed to be necessary for the long term development and success of the project. There are a total of 0.049% of the total Noso produced, the equivalent of 10303.9073 Noso premined in block zero. See Noso production reward schedule below:

From	To	Blocks	Reward/Block	Coins/period	Total supply
0	1	1	10303.90730000	10303.90730000	10303.90730000
1	209999	209999	50.00000000	10499950.00000000	10510253.90730000
210000	419999	210000	25.00000000	5250000.00000000	15760253.90730000
420000	629999	210000	12.50000000	2625000.00000000	18385253.90730000
630000	839999	210000	6.25000000	1312500.00000000	19697753.90730000
840000	1049999	210000	3.12500000	656250.00000000	20354003.90730000
1050000	1259999	210000	1.56250000	326125.00000000	20682128.90730000
1260000	1469999	210000	0.78125000	164062.50000000	20846191.40730000
1470000	1679999	210000	0.39062500	82031.25000000	20928222.65730000
1680000	1889999	210000	0.19531250	41015.62500000	20969238.28230000
1890000	2099999	210000	0.09765625	20507.81250000	20969746.09480000

2100000	2309999	210000	0.04882812	10253.90520000	21000000.00000000

The premined Noso is used for promotion of the project through gifts, prizes, contests and activities. This promotion is to be conducted through the different channels and social networks (blog, Discord, Facebook, Twitter, etc.) to achieve maximum diffusion and reach the largest number of users.

As depicted in **Figure 1**, the production of Noso is scheduled to continue throughout much of the 21st century, slowly reducing every 210,000 blocks until 2065.

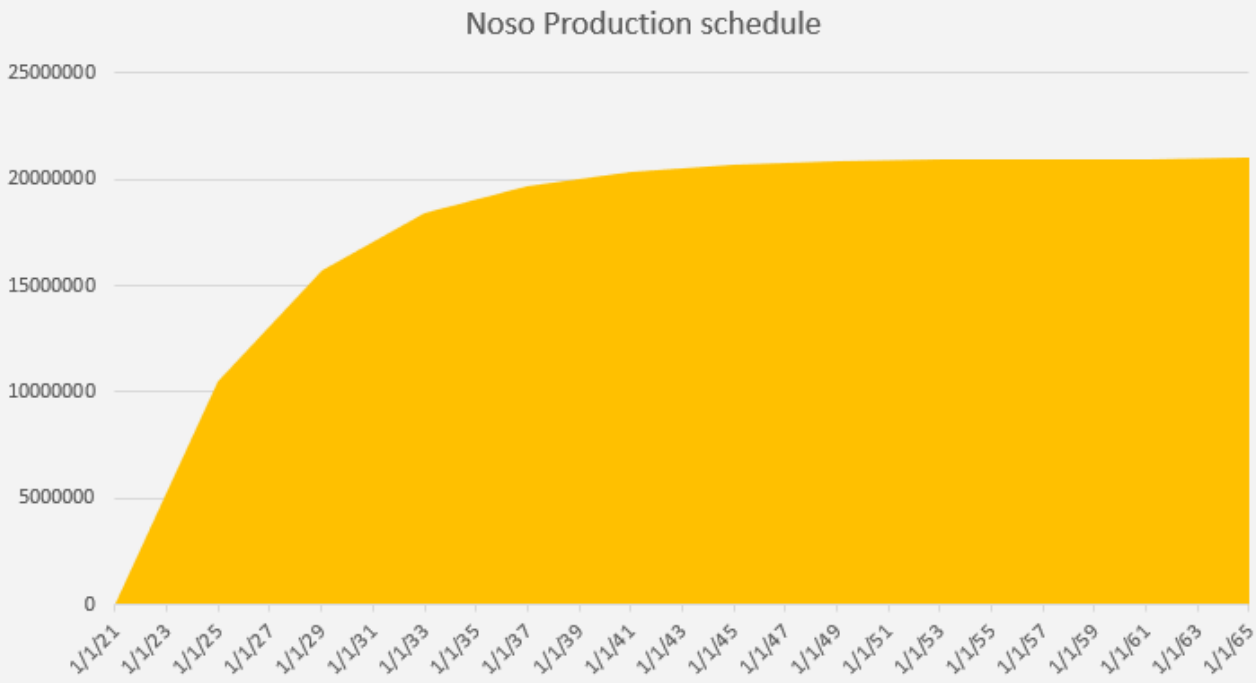

Figure 1

NOSO VOTING SYSTEM

Built into Noso protocol is a Governance Token (GVT) poll management system (under development). This system allows GVT owners to create project related polls for consideration which are voted upon by other GVT owners. In order to participate in the voting process, an individual must acquire through purchase or trade, a Noso Governance Token (GVT). These tokens are available to anyone wanting to participate in the voting process given availability.

Governance Tokens have the following specifications:

- Initial 100 GVTs have been minted
- GVTs are transferable
- 1 GVT = 1 vote
- All GVT owners can start polls

- Polls with +75% of approbation will be mandatory; +50% will be recommended
- New GVTs are minted every year only (on demand, after government poll approbation)
- GVTs can be banned by other GVTs owners in certain circumstances. When banned, the GVT owner becomes suspended from creating new polls or participating on existing ones for 90 days.
- Each GVT can only run one poll at a time
- GVTs rules itself can be adjusted by a GVT votation.

Example GVT poll: Should NOSO reduce the PoW reward to 5%?

NOSO BLOCKCHAIN DEVELOPMENT

Noso takes a functional approach to development. From real world examples, use cases and community suggestions, it strives to develop a blockchain that can be deployed and widely used for practical applications. Noso welcomes everyone to participate in shaping its roadmap and technology by contributing directly to code development, proposing new ideas, or submitting comments and suggestions.

If you have the skillset and would like to get started as an independent Noso developer, you must first obtain the consensus tree from the Noso project repository here (new unit nosoconsensus): https://github.com/Noso-Project/libraries/blob/main/nosoconsensus.pas The nosoconsensus unit keeps the Noso mainnet consensus always in background and doesn't use app main threads.

To build blockchain related apps (like nosonode), you must manage the three supports (block, summary and headers). These can be obtained here: https://github.com/Noso-Project/nosod/blob/main/nosounit.pas

To build external apps (like nosolite), you need to manage the valid protocol commands. These can be found here: https://github.com/Noso-Project/NosoNode/blob/main/mpprotocol.pas#L316

Anyone wanting to participate as part of the project should reach out to the Noso community through social media channels.

NOSO TRANSACTIONAL OPERATION

Key Generation

For each new address created, a new ECDSA key pair is generated. Private keys are stored only in a user's wallet, while the public key using MD160 and SHA-256 hash, yields a valid Noso address like the following:

N2kFAtGWLb57Qz91sexZSAnYwA3T7Cy

This public address has the following characteristics:

- Length: 30 or 31 Characters.
- The first character is always a capital 'N'
- The remaining characters are on base 58 (same as Bitcoin) Range: 0-9,A-Z, a-z. Except number zero (0), capital letters O and I and letter l
- The last 2 characters work as checksum, to avoid miss typing errors.

Your public address is what is shared to send and receive transactions. Private keys should never be shared and doing so could result in complete loss of your Noso coins.

Noso Account Summary (Addresses):

The file **sumary.psk** contains information including all existing addresses in the protocol, available balances and the current status of the blockchain. This file is updated with each newly created block and contains all information needed to start operations in the network. This means new users do not need to download the whole blockchain to start using their portfolio, create new blocks or verify new transactions in the network. Because the size of this file can increase unpredictably if a user were to immediately register large numbers of new addresses, the protocol summary only includes addresses that have already received a transaction or have created a block. This limits the sumary.psk file size to that of only addresses in use.

Accounts summary Optimization

Keeping the protocol efficient is a fundamental part of the project. To achieve this, the following measures are included.

1. **Reference** (from the Spanish: concept) is an alphanumeric chain added in each transfer. This field can contain descriptive information which the sender can include for the receiver. This reference is included in the digital transfer signature and helps to avoid double-spending situations. The length of the reference has a maximum of 40 characters and is not case sensitive.
2. **List** of all transactions are included in the block.
3. **Double spend control**. In the Noso protocol, double-spending control is carried out by verifying the transaction hash and signature, and the previous balance of the sending address and pending transactions. Since all transactions have a timestamp

incorporated, a signed transaction will only be valid for the current block or the next one. Unlike Noso's predecessors, this means Noso is not able to perform offline transactions, however, it allows faster calculation and verification of transactions and the virtual elimination of orphan blocks. Also, a transaction can not be duplicated in the same block or exist in the previous block. Therefore in order for a transaction to be accepted, its creation timestamp must be higher than than the start of block X-1 or it will be automatically discarded.

4. **Network time**. Given the nature of the Noso algorithm, precision in the synchronization of nodes is very important. Although it does not require absolute accuracy, maintaining very similar time is necessary. This is why Noso mainnet uses reliable public NTP servers, synchronizing node time before comparing with other members on the network. Although the difference in latency should never exceed two seconds, the protocol establishes a five seconds maximum time allowed to accept transactions from a pair. This time synchronization prevents late and malicious retransmission of duplicate operations. All operations within the Noso network use the UTC reference time.
5. **Orphan Blocks.** The Noso protocol is designed to prevent the creation of orphaned blocks and if one is created, to prevent its propagation through the network. This is achieved by keeping all pending transactions uniform and modifying their structure only if a transaction update will not be included in the next block; allowing the network enough time to re-verify all subsequent transactions. This means that if one or more transactions are generated shortly before the creation of a new block (60 seconds or less), those transactions will be included in the subsequent block.
6. **Minimum Fees.** The following minimum fees are established by the protocol. All orders have a minimum per-order fee of 0.00000010 Noso and a minimum transaction fee of 0.00000010 Noso. These fees help to avoid saturation of the network with micro transactions.

Noso network files

In addition to sumary.psk, the following files each play their own role and are necessary to maintain smooth network operation and recovery. All network nodes maintain the following files in their Noso directory.

- /NOSODATA/BLOCKS/10000.blk : Block files directory location
- /NOSODATA/LOGS/exceptlog.txt : Exceptions logs
- /NOSODATA/SUMMARKS/GVTS/10000.bak : GVTs status at each block (necessary for recovery
- /NOSODATA/SUMMARKS/10000.bak : Summary status at each block. (necessary for recovery).
- /NOSODATA/UPDATES/*.zip : Downloaded updates and compressed binaries

- /NOSODATA/UPDATES/Noso.exe : Last binary uncompressed
- /NOSODATA/advopt.txt : User node specific options file
- /NOSODATA/blchhead.nos: Blockchain headers; this is the block chain merkle root
- /NOSODATA/botdata.psk : Local banned ips list
- /NOSODATA/logs/eventlog.txt : Nosodebug log
- /NOSODATA/gvts.psk: Current state of gvts
- /NOSODATA/gvts.psk.bak : Security backup of gvts.psk file
- /NOSODATA/masternodes.txt : Masternodes of the last mined block
- /NOSODATA/nosocfg.psk : Status of the mainnet critical variables
- /NOSODATA/sumary.psk : Summary status after last built block. (AKA current summary state)
- /NOSODATA/sumary.psk.bak: backup of the summary before the last built block
- /NOSODATA/sumary.zip : Current summary state zipped for sending purposes
- /NOSODATA/wallet.psk : The public/private keys file
- /NOSODATA/wallet.pwk.bak: Backup of the wallet.psk

Sending and Receiving Noso coins

When sending Noso, a sender's wallet will locate an address or group of addresses to cover the order amount plus fees needing to be sent. Each sending address in the order generates a TRANSFER; therefore an order will ALWAYS need one or more transfers to be filled (Please note, each order can have multiple senders but only one receiver).

Each transfer contains the source address and how much that address is contributing to the payment and fees of the order. Once all transfers are generated, the protocol builds an order and generates a unique orderID. This orderID is also included in the transfers data and can not be double spent.

Example:

Sender1 wants to send 100 Noso + 1 Noso for fees to a target address.

Sender1 has 3 addresses in his wallet with funds, each containing 40 Noso.

The first, 3 transfers are generated.

Transfer1 : AddressA send 40 Noso, pays 0 fee

Transfer2 : AddressB send 40 Noso, pays 0 fee

Transfer3 : AddressC send 20 Noso, pays 1 fee

The order is then created, generating a unique OrderID for all of them (This process works like a package for a group of transfers). The order also contains the receiving address,

the total amount to be credited and fees paid.

OrderID contains: Transfer 1 , Transfer 2 and Transfer 3.

Even though transfers are what really "move" funds from one address to another, a transfer always needs to be part of one order to be processed.

At block creation, orders are processed always sequentially. This means one after another, no matter how many transfers each order has. When transfers signatures are validated, the order is considered legit, funds are taken from senders and credited to the receiver.

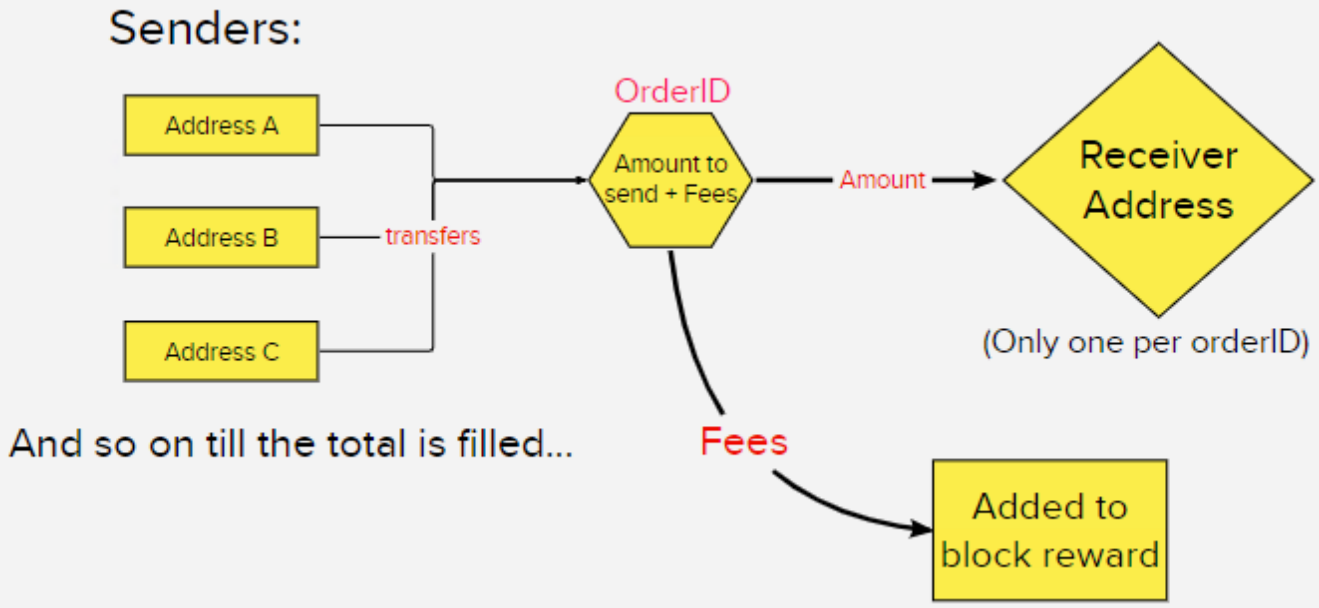

Figure 2: Noso transaction flow

NOSO BLOCK CREATION

The block creation process contains four distinct functions.

1. **Processing regular orders**

"Regular Orders" are existing funds sent to and from Noso addresses. The following code shows how regular orders are processed.

```
// Processs pending orders
  EnterCriticalSection(CSPending);
  BeginPerformance('NewBLOCK_PENDING');
  ArrayLastBlockTrxs := Default(TBlockOrdersArray);
  ArrayLastBlockTrxs := GetBlockTrxs(MyLastBlock);
```

```
   ResetBlockRecords;
   for contador := 0 to length(pendingTXs)-1 do
      begin
      // Version 0.2.1Ga1 reverification starts
      if PendingTXs[contador].TimeStamp < LastBlockData.TimeStart then
         continue;
      //{
      ExistsInLastBlock := false;
      for count2 := 0 to length(ArrayLastBlockTrxs)-1 do
         begin
         if ArrayLastBlockTrxs[count2].TrfrID = PendingTXs[contador].TrfrID then
            begin
            ExistsInLastBlock := true ;
            break;
            end;
         end;
      if ExistsInLastBlock then continue;
      if PendingTXs[contador].TimeStamp+60 > TimeStamp then
         begin
         if PendingTXs[contador].TimeStamp < TimeStamp+600 then
            insert(PendingTXs[contador],IgnoredTrxs,length(IgnoredTrxs));
         continue;
         end;
      if PendingTXs[contador].OrderType='CUSTOM' then
         begin
         OperationAddress := GetAddressFromPublicKey(PendingTXs[contador].sender);
         if
IsCustomizacionValid(OperationAddress,PendingTXs[contador].Receiver,numero) then
            begin
            minerfee := minerfee+PendingTXs[contador].AmmountFee;
            PendingTXs[contador].Block:=numero;
            PendingTXs[contador].sender:=OperationAddress;
            insert(PendingTXs[contador],ListaOrdenes,length(listaordenes));
            end;
         end;
      if PendingTXs[contador].OrderType='TRFR' then
         begin
         OperationAddress := PendingTXs[contador].Address;
         if
SummaryValidPay(OperationAddress,PendingTXs[contador].AmmountFee+PendingTXs[contador].
AmmountTrf,numero) then
            begin
            minerfee := minerfee+PendingTXs[contador].AmmountFee;

CreditTo(PendingTXs[contador].Receiver,PendingTXs[contador].AmmountTrf,numero);
            PendingTXs[contador].Block:=numero;
            PendingTXs[contador].sender:=OperationAddress;
            insert(PendingTXs[contador],ListaOrdenes,length(listaordenes));
            end;
         end;
```

```
      if ( (PendingTXs[contador].OrderType='SNDGVT') and ( PendingTXs[contador].sender
= AdminPubKey) ) then
         begin
         OperationAddress := GetAddressFromPublicKey(PendingTXs[contador].sender);
         if GetAddressBalanceIndexed(OperationAddress)<
PendingTXs[contador].AmmountFee then continue;
         if
ChangeGVTOwner(StrToIntDef(PendingTXs[contador].Reference,100),OperationAddress,Pendin
gTXs[contador].Receiver)=0 then
            begin
            minerfee := minerfee+PendingTXs[contador].AmmountFee;
            Inc(GVTsTransfered);
            SummaryValidPay(OperationAddress,PendingTXs[contador].AmmountFee,numero);
            PendingTXs[contador].Block:=numero;
            PendingTXs[contador].sender:=OperationAddress;
            insert(PendingTXs[contador],ListaOrdenes,length(listaordenes));
            end;
         end;
      end;
```

2. Processing PoS payments (depreciated)

```
           Depreciated
```

3. Processing Masternode (MN) payments

"MN payments" are newly minted coins sent to Masternode operators. See below "Masternode Network" for details on Masternode payments.

4. Creating block headers and saving the block

The final steps leading up to block creation. See the code below on how block headers are created.

```
   // Definir la cabecera del bloque *****
    BlockHeader := Default(BlockHeaderData);
    BlockHeader.Number := Numero;
    BlockHeader.TimeStart:= StartBlockTime;
    BlockHeader.TimeEnd:= timeStamp;
    BlockHeader.TimeTotal:= TimeStamp - StartBlockTime;
    BlockHeader.TimeLast20:=0;//GetLast20Time(BlockHeader.TimeTotal);
    BlockHeader.TrxTotales:=length(ListaOrdenes);
    if numero = 0 then BlockHeader.Difficult:= InitialBlockDiff
    else if ( (numero>0) and (numero<53000) ) then BlockHeader.Difficult:= 0
    else BlockHeader.Difficult := PoSCount;
    BlockHeader.TargetHash:=TargetHash;
    //if protocolo = 1 then BlockHeader.Solution:= Solucion
    BlockHeader.Solution:= Solucion+' '+GetNMSData.Diff+' '+PoWTotalReward.ToString+'
'+MNsTotalReward.ToString+' '+PosTotalReward.ToString;
    if numero = 0 then BlockHeader.Solution:='';
```

```
   if numero = 0 then BlockHeader.LastBlockHash:='NOSO GENESYS BLOCK'
   else BlockHeader.LastBlockHash:=MyLastBlockHash;
   if numero<53000 then BlockHeader.NxtBlkDiff:=
0{MNsReward}//GetDiffForNextBlock(numero,BlockHeader.TimeLast20,BlockHeader.TimeTotal,
BlockHeader.Difficult);
   else BlockHeader.NxtBlkDiff := MNsCount;
   BlockHeader.AccountMiner:=Minero;
   BlockHeader.MinerFee:=MinerFee;
   BlockHeader.Reward:=GetBlockReward(Numero);
   // Fin de la cabecera -----
   // Guardar bloque al disco
   if not
GuardarBloque(FileName,BlockHeader,ListaOrdenes,PosReward,PosCount,PoSAddressess,
                        MNsReward, MNsCount,MNsAddressess) then
      AddLineToDebugLog('exceps',FormatDateTime('dd mm YYYY HH:MM:SS.zzz', Now)+' ->
'+'*****CRITICAL*****'+slinebreak+'Error building block: '+numero.ToString);
```

MASTERNODE NETWORK

Masternodes are servers running on the Noso P2P network providing clients the ability to facilitate the efficient propagation and distribution of messages throughout the network. Masternode operators must provide collateral to operate a node and in turn earn payment for the services provided while stabilizing and reducing volatility of the currency.

Masternode operators are required to maintain a Noso balance of 10,500 coins and a server running NosoNode software on a 24-hour basis.

The following diagram **Figure 3**, shows reachable Noso masternodes over a typical 30 day period (12/1/2022 - 1/1/2023).

Figure 3: Noso Masternodes count

Based on the Noso protocol, Masternodes create a block every 600 seconds. When each block is created, a block reward is split between masternodes and project funds.

The masternodes portion of the block reward is processed in the following way. Unlike regular orders containing a sender, receiver, amount and signature, masternode payments contain only the list of addresses to be paid (receiver), and the payment amount. The sender is not needed as coins are derived from coinbase and send amounts are automatically validated by other masternodes.

When calculating masternode payments, A count of all active masternodes is performed and verified. With the current Masternode count, an equal payment amount is determined for all active nodes. Upon verification, masternode addresses and payment amounts are saved to disk and rewards are distributed to all active nodes recorded in the last block.

Masternode Processing code example:

```
  // Masternodes processing
   BeginPerformance('NewBLOCK_MNs');
   CreditMNVerifications();
   MNsFileText := GetMNsAddresses();
   SaveMNsFile(MNsFileText);
   ClearMNsChecks();
   ClearMNsList();
   if numero >= MNBlockStart then
      begin
      SetLength(MNsAddressess,0);
      Contador := 1;
      Repeat
         begin
         ThisParam := Parameter(MNsFileText,contador);
         if ThisParam<> '' then
            begin
            ThisParam := StringReplace(ThisParam,':',' ',[rfReplaceAll]);
            ThisParam := Parameter(ThisParam,1);
            SetLength(MNsAddressess,length(MNsAddressess)+1);
            MNsAddressess[length(MNsAddressess)-1].address:=ThisParam;
            end;
         Inc(contador);
         end;
      until ThisParam = '';

      MNsCount := Length(MNsAddressess);
      MNsTotalReward := ((GetBlockReward(Numero)+MinerFee)*GetMNsPercentage(Numero))
div 10000;
      if MNsCount>0 then MNsReward := MNsTotalReward div MNsCount
      else MNsReward := 0;
      MNsTotalReward := MNsCount * MNsReward;
      For contador := 0 to length(MNsAddressess)-1 do
         begin
         CreditTo(MNsAddressess[contador].address,MNsReward,numero);
         end;
      EndPerformance('NewBLOCK_MNs');
```

```
end;// End of MNS payment procecessing
```

Shared Masternode Contracts

Shared Masternode Contracts provide a mechanism for enhanced community participation in node hosting within the Noso network. With the introduction of shared nodes, participants can collaborate and pool their resources, enabling a more decentralized and inclusive environment.

Masternodes within the Noso network are categorized into two types: Solo Nodes and Shared Nodes. Solo Nodes allow individual participants to host and maintain their own Masternode on their own hardware. Shared Nodes, on the other hand, function similarly to Solo Nodes but with the added characteristic of funds being shared among multiple Noso addresses.

To initiate a Shared Node, interested individuals can send a "SharedNode Contract request" to mainnet. This request requires a fee, typically around 0.20 Noso, and serves as a public announcement of their intention to host a shared node. The request indicates their desire to seek funding from the community to support the shared address. Additionally, the SharedNode Contract request includes specifications such as the time duration of the contract in blocks, the fee percentage that the contract creator will retain for hosting and maintaining the node instance, and the minimum participation required, which aligns with or exceeds the protocol's minimum allowed threshold.

Any participant who wishes to join a Shared Node Contract can do so by submitting a "signed agreement" to the contract. This agreement is free of charge for participants. However, by joining, they agree to freeze the coins in their selected address until the Shared Node Contract reaches its expiration. It's important to note that participants maintain ownership of their coins throughout the contract period.

The distribution of funds within the Shared Node registered address occurs automatically at the protocol level. This ensures a fair and transparent process that cannot be manipulated by any individual, including the Shared Node creator. The protocol distributes the coins held in the shared address proportionally among all participants, including the contract creator. This distribution mechanism provides a high level of security, preventing scams or unauthorized withdrawals.

Within the Shared Node Contract system, two key roles exist: the Shared Node Host and the Shared Node Earner. The Shared Node Host is the individual who creates the Shared Node Contract, pays the creation fee, and operates the node instance on a VPS or their preferred location.

To prevent situations where a contract becomes overfilled, an "overfilled" protection mechanism is in place to restrict additional participants from joining the contract if their

contributions would cause the total funds to exceed the collateral size by more than 10%. This safeguard ensures a fair and balanced distribution of rewards among participants.

Shared Node Contracts facilitate increased community engagement and participation in node hosting. By allowing participants to collaborate and pool their resources, Shared Nodes contribute to a more decentralized and inclusive network environment. These mechanisms provide transparency, security, and fairness within the Shared Node Contract system, fostering a vibrant and robust ecosystem.

NOSO NETWORK FLOW

The TCP handshake

Masternodes connect to one another using an always-on TCP stream connection.

- Masternodes will send an initial Hello line, then continue sending a Ping line every 5 seconds thereafter.
- These messages are composed of a string with space separated values terminated by an End-Of-Line(EOL) sequence.

The Hello

This is a space separated string with the following data/fields:

1. The string PSK
2. The target IP address: String
3. The version of the software: String
4. The current UTC Unix timestamp: Integer

Format string: PSK %s %s %d

The Ping

This is a space separated string with the following data/fields:

1. The string PSK
2. The current protocol version 2: Integer
3. The version of the software: String
4. The current UTC Unix timestamp: Integer
5. The string $PING: String
6. The current amount of connections: Integer
7. The current block number(Default: 0): Integer
8. The current block hash(Default: 4E8A4743AA6083F3833DDA1216FE3717): String
9. The hash of the NOSODATA/sumary.psk file(Default: D41D8CD98F00B204E9800998ECF8427E): String

10. The amount of pending orders(Default: 0): Integer
11. The hash of the NOSODATA/blchhead.nos file(Default: D41D8CD98F00B204E9800998ECF8427E): String
12. Status of the connection: Integer
    - 0: Disconnected
    - 1: Connecting
    - 2: Connected
    - 3: Updated
13. Connection port: Integer
14. First five characters of the hash of the NOSODATA/masternodes.txt file: String
15. The amount of masternodes: Integer
16. NMsData difference: String
17. The amount of checked masternodes: Integer
18. Hash of the NOSODATA/gvts.psk file: String
19. First five characters of the hash of the CFGs: String

Format string: PSK %d %s %d $PING %d %d %s %s %d %s %d %d %s %d %s %d %s %s

4E8A4743AA6083F3833DDA1216FE3717 is the value of the hash for Block 0(zero).

D41D8CD98F00B204E9800998ECF8427E is the value of md5 on an empty string.

The Pong

This is a space separated string with the following data/fields:

1. The string PSK
2. The current protocol version 2: Integer
3. The version of the software: String
4. The current UTC Unix timestamp: Integer
5. The string $PONG: String
6. The current amount of connections: Integer
7. The current block number: Integer
8. The current block hash: String
9. The hash of the NOSODATA/sumary.psk file: String
10. The amount of pending orders: Integer
11. The hash of the NOSODATA/blchhead.nos file: String
12. Status of the connection: Integer
    - 0: Disconnected
    - 1: Connecting
    - 2: Connected
    - 3: Updated
13. Connection port: Integer

14. First five characters of the hash of the NOSODATA/masternodes.txt file: String
15. The amount of masternodes: Integer
16. NMsData difference: String
17. The amount of checked masterNodes: Integer
18. Hash of the NOSODATA/gvts.psk file: String
19. First five characters of the hash of the CFGs: String

Format string: PSK %d %s %d $PONG %d %d %s %s %d %s %d %d %s %d %s %d %s %s

4E8A4743AA6083F3833DDA1216FE3717 is the value of the hash for Block 0(zero).

D41D8CD98F00B204E9800998ECF8427E is the value of md5 on an empty string.

It is important to remember that PING is identical to PONG in format with the only difference being that PING requires an answer (PONG) from the peer, while PONG does not.
All received pings are replied to with a pong. The ping not only checks if a connection is alive, it also updates necessary data from that peer.
A peer will verify how long it has been since the last ping from all connected peers and if the last ping from a peer is more than 5 seconds old, it sends another ping. The PING-PONG system keeps all nodes updated to the last 5 seconds with all connected peers and allows for fast propagation of information in mainnet.

Ping/Pong code example:

```
function GetPingString():string;
var
  Port : integer = 0;
Begin
if Form1.Server.Active then port := Form1.Server.DefaultPort else port:= -1 ;
result :=IntToStr(GetTotalConexiones())+' '+
         IntToStr(MyLastBlock)+' '+
         MyLastBlockHash+' '+
         MySumarioHash+' '+
         GetPendingCount.ToString+' '+
         MyResumenHash+' '+
         IntToStr(MyConStatus)+' '+
         IntToStr(port)+' '+
         copy(MyMNsHash,0,5)+' '+
         IntToStr(GetMNsListLength)+' '+
         GetNMSData.Diff+' '+
         GetMNsChecksCount.ToString+' '+
         MyGVTsHash+' '+
         Copy(HashMD5String(GetNosoCFGString),0,5);
End;
```

PROJECT FUNDS

Project Funds were introduced at block 88406 as a method to compensate those who help to make the project better in the following categories.

1. Project coding / protocol
2. Project documentation
3. Project promotion
4. Other impactful contributions to the project not mentioned

The way that project funds works is, 10% of each minted block reward is forwarded to the special Project funds address (NpryectdevepmentfundsGE). This special address contains no keys and is only accessed by governance directive. This means a majority vote must be held to release funds for each project contribution.

The project funds order is a "special" order which is included at the end of every "regular orders'' list. These "special" orders are generated independently by every masternode when each block is created and therefore are never propagated throughout the network like regular orders.

Project funds payment code example:

```
   // Project funds payment
   if numero >= PoSBlockEnd then
      begin
      DevsTotalReward := ((GetBlockReward(Numero)+MinerFee)*GetDevPercentage(Numero))
div 10000;
      DevORder := CreateDevPaymentOrder(numero,TimeStamp,DevsTotalReward);
      CreditTo('NpryectdevepmentfundsGE',DevsTotalReward,numero);
      insert(DevORder,ListaOrdenes,length(listaordenes));
      end;
   if GVTsTransfered>0 then
      begin
      SaveGVTs;
      UpdateMyGVTsList;
      end;
   TRY
      SetLength(PendingTXs,0);
      PendingTXs := copy(IgnoredTrxs,0,length(IgnoredTrxs));
   EXCEPT on E:Exception do
      begin
      AddLineToDebugLog('exceps',FormatDateTime('dd mm YYYY HH:MM:SS.zzz', Now)+' ->
'+'Error asigning pending to Ignored');
      end;
   END; {TRY}
   SetLength(IgnoredTrxs,0);
```

```
EndPerformance('NewBLOCK_PENDING');
LeaveCriticalSection(CSPending);
```

FORMAL DEFINITIONS

- **Block chain**: Data source which keeps information updated within the protocol. Each block contains all transactions from the moment a previous block was created until the moment the next block is created.

AREAS OF CONSTANT IMPROVEMENT

- Reducing energy consumption of the network
- Increasing transactions per second from 50tx/s to 100tx/s, then 1000tx/s and beyond
- Reducing blockchain sync times and resource utilization
- Reducing blockchain size
- Increasing reliability and security of the network
- Regular audits and updates of the core cryptography to negate the benefits of quantum computations.
- Continuous enhancements and support of Noso Wallets, and other applications released by the Development team
- Continuous improvements to Whitepaper and supporting documentation
- Integration of NoBiEx (Noso Built In Exchange) a full peer-to-peer exchange that will provide liquidity that Nosocoin needs without the ever need of kyc and eliminates the 3rd party like binance,coinbase…etc
- Protocol 3 development to include smart contracts
- Protocol 3 transfer confirmations reduced to 60 seconds

REFERENCES:

https://developer.bitcoin.org/reference/block_chain.html

https://bitcoin.org/bitcoin.pdf

https://github.com/Noso-Project/NosoNode/blob/main/mpblock.pas#L133

https://github.com/Noso-Project/NosoNode/blob/main/mpblock.pas#L224

https://github.com/Noso-Project/NosoNode/blob/main/mpblock.pas#L245

https://github.com/Noso-Project/NosoNode/blob/main/mpblock.pas#L300

https://github.com/Noso-Project/NosoNode/blob/main/mpblock.pas#L199