TODA Protocol - A Ledgerless Blockchain

A new approach for a Fully Distributed and Decentralized Protocol

Intro:

There is currently a need for a strong governance *(decentralized)* value exchange platform that also uses distributed computing so it scales efficiently to millions of transactions per second, powering billions of devices while increasing security and minimizing transaction fees *(to almost zero)*. The TODA protocol is designed to meet those needs, is fully distributed (BFT, Repli-Max x32, *efficient yet redundant, reliable and secure)*, has a governance model which becomes more and more decentralized as it grows *(strong governance)*, and is optimized for mobile devices without the need for any centrally controlled system. It can also run on combination of nano-micro cloud instances, one representing each mobile device. *// this step might be necessary at the very beginning, so it's highly recommended that implementation details take that into effect. See PoAW for details of how more than necessary cloud power does not make any difference, we call that CMR (Centralized Mining Resistant)*

Design overview:

In a fully distributed and decentralized system, nodes will need a map to efficiently navigate the network and locate objects. A virtual binary tree, we call it the Todatree, specifically a fully saturated BStree structure is a virtual structure on the protocol level that contains a reference to every object in the entire system that exist, will exist and objects that move from one place in the tree to another, they move using unique IDs from the tree as their identifiers and occupy points in the tree that are also unique. Given it is fully distributed, the reference is actually from the device pointing to the device and all the potential nodes that it could potentially interact with. We use it because it is extremely fast to compute at each nodes level to locate a certain leaf's parent in a certain branch etc. and it is easy to any computer scientist to comprehend it and imagine its navigation and the possibilities of things that can be done deterministically. *// This abstraction is needed because we envision at least 7 different cs disciplines to work with TODA protocol, so it's important that we not only make it easy for machines, but as easy as we could to the subject matter experts. At any point some deployments may succeed without this specific abstraction while still preserving the overall protocol instructions.*

The smallest Unit is at the leaf's level we call it TODAQ or Quark, and the smallest unit that can not be divided by users but it can be transferred in its wholly by users we call it TodaNote, tNote, TODAUnit or even Toda File because it is a file. It has an atomic structure in a sense that it has a unique number through its life and quarks actually form it within that tree structure, each quark has a unique number and can contribute along with a total of 2^32 Quarks to form one single unique tNote.

// similar to atoms in the universe made out of quarks and other sub-atomic elements, but you can not send half an atom on its own. So one set of 2^32 quarks can form one specific tNote with one unique number and not any other tNote
One application of TODA protocol a value to be exchanged similar to a currency, and the provenance of each unit/tNote is from a sub-branch at level 96 the Provenance Wallet *(technically 2^63 tNotes is the maximum limit to be initiated into the system because tNotes are at level 32)* - A tNote can travel within the Todatree and have a new location every block only when transacted. Every movement builds a chain of Wallets edited inside the tNote file *(also at level 96 in todatree, some implementation details may suggest level 64 which aren't concerned with the quarks and creation of tNotes).*

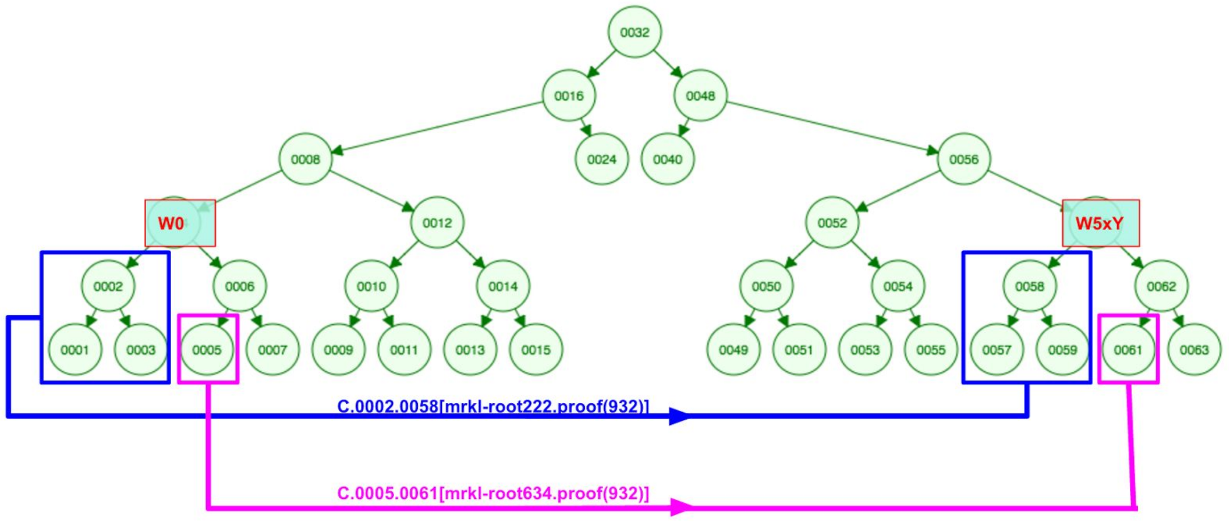
- **Atomic tNotes:** *// transaction fees and creation of new tNotes -*
  - Each unit/tNote contains its own history as a wallet chain of past transactions. If UserA sends UserB tNote C115, from the user's perspective, the tNote would look like this: C115_W(UserA).BlockNumber1111.MerkleProof1111.7854_to_W(UserB).BlockNumber88888.MerkleProof88888.569 // To the machine, each tNote can only occupy certain location in the wallet, that location is basically a branch and therefore has a unique number in the entire system. Also tx *sigs are inside MerkleProof*
  - TodaNote is the smallest unit that can be transacted by users *//In early implementations we are recommending the minimum suggested tNotes to be transacted to be 16, this increases the cost of attack vectors and reduces the success rate of attacks significantly that way tNotes include the merkle proof of theirs and that of 15 others minimum.*
  - TODAQ or Quark is the smallest unit non-divisible, it is mainly used for exchange rate of 0.03% for the payer cost, 0.03% for the payee and 0.04% of new quarks that are issued every transaction that can be sent to the transaction wallet of users who win the PoAW. 2^32 quarks = one tNote.
  - The Quarks are actually the leafs of the branch TodaNote in todatree*
  - Quarks are managed within one tNote until filled up, so the management of that is basically an increment to the tNote extension. Say for example UserA owns 99.9% fractions of C113 that were gained by validating other tNotes in transactions, it would be represented by C113.4290672329_UserA, then if UserA validates another tNote being transacted and wins the transaction fee as per PoAW then user A Qualifies to collect 0.1% in transaction fee on that tNote which is 4294968 Quarks at that point the C113 is filled up *(because 4290672329+4294968=4294967297 which is 2^32)* and now can be spent by UserA. Similarly if instead UserA sends 100 tNotes to UserB, UserA will have to pay the 0.03% or *(100*0.0003*4294967297=128849019)* Quarks which will reduce C113 from 4390672329 to 4161823310. UserA will then have

C113.4161823310. Another implementation would be using a hashcash like of next block's merkleroot concat of existing transaction hash to charge 1tNote only 0.03% of the times which effectively be the same results but don't have to deal with splitting of the tNote at the user level. The Toda Protocol CORE Implementation uses this method.

- Basically, because it is 0.04% of new tNotes are created per tNote movement, a tNote will have to change ownership 2500 times so its impact aggregate is creating 1 new tNote

- **Todatree**
  - Like a BST of height 256 *(Suggested tree model is based on fully saturated and balanced BST)*. The leftmost branch of level 160 can be viewed as a tree itself, and transactions within it can be processed more efficiently than in the full tree, but we keep 256 by design for ease of future expansion.

Todatree simplified 256 to 5



This tree is used as a map so nodes know how to navigate the system and where to expect objects to be. *// This tree has more reference points than the estimated number of atoms in the universe we know of*

- We call this tree the Todatree. We refer to level 96 as t96 and it is the level that represents the nodes/wallets. *// some implementations suggestions Wallets are at 64*
- At the very beginning, the very first 200 wallets on the left, have the provenance of all the tNotes *// other branches can have units for smart contracts or whatnot*
- Quarks are at the leafs level t0 of that wallet, However not all quarks have same value, they all depend on the provenance Wallet, originally we recommend denominations to be represented in left or right wallets such as W100 is where the the TODA Notes, 20 to the left would 0.01TODA 20 to the right would be

100TODAs, 30 would 1000TODAs This makes easier to send small or large amounts, can be computed in a combination that will always be minimum 17 tNotes during one transaction. The average amount of files per transaction will depend on the economy using it, but will likely be between 20 and 30.

- tNotes have a unique number for the life of them. Their number is that of the branch they come from, and their movements are branches they occupy belonging to other wallets, hence wallet chain is within each and every tNote is calculated at the system level as it is better from usability perspective to show wallet chain than point occupied chain, especially if your wallet remembers other wallets and gives them names. Think of phone numbers, you remember the name of someone but they could have several phone numbers.
- Every block, the entire provenance range of W200 branches move to the right of the first wallet range of 200 by 200 every block. The reason for that is so when looking at any tNote number you can tell its provenance block by conducting a quick binary computation, so the provenance W of a tNote is linearly related to the block# so the calculation would be: Provenance wallet = (W(2^95 + (block# * (2^96))) - *//This is important to prevent tNote creation/activation in some attack vectors*
- **Merkle Tree** //see appendix
    - Generated every T seconds
    - The root hash is generated by every active validating node in the network shared across the network
    - Every tNote is coupled with another tNote to potentially form L1 *L1 for Level one)* of new merkle tree *(Merkle tree and Todatree are not the same, merkle tree is another dimension that gets created every block, while todatree is static for the life of the platform)* utilizing the previous Merkle root in a function to pseudorandomly select a range for the coupling of tNotes along with the nodes/managers of those tNotes *//Which manager coupling with which other set of managers. Hint: Each node is assumed to be selfish and winner ones are the ones capable of evolving, by learning who to pick for coupling along with other elements, we expect toda nodes to evolve very rapidly given the incentives, social scalability will likely be the bottleneck of distribution as people will not easily trust a new software or even an update, however the incentive is there to any software developer to build better algorithms and find a way to get them adopted.*
    - Only managers with at least one tNote that is being transacted in the current block can be going up to L2 and all the way up to Merkle root, in fact if not transacting any tNote during a block, the node is not incentivised nor permitted to find anyone to couple with at L2

  - Basically every node when acting as manager is able to run the pseudorandom function that provides the coupling for each and everyone of its tNotes that it is managing.
  - The non-transacting tNote manager will be relying on the transacting tNotes managers to get back with the new merkle proof
    - If a manager *(hint: every wallet acts as a manager to some other tNotes each block see Beacon)* does not have any tNotes transacting and therefore not incentivized nor authorized/expected to participate in building the Merkle tree, it will need to rely on the other nodes to relay what was the last Merkle root. The same applies to dormant or disconnected nodes, when they rejoin the network using gossip like protocol *(whatever you use make sure if it isn't resilient to attacks on its own to utilize the merkle proof to be verified by nodes if they were to propagate it)*
  - Every participating wallet must have all Merkle roots in the system along with last tNote assigned and lasr wallet assigned during each Merkle root *// that's basically the only universally shared data if a wallet doesn't have this data, it will have hard time validating instantaneously receipt of transactions as it would be able to compare merkleproofs against historic merkleroots*
- **Beacon**
  - The Merkle root of last block is used as a beacon in existing block to generate pseudo-randomness that is deterministic and universal.
    - For example, the Merkle root is used in a function to tell each wallet which tNotes to manage during the existing block
    - To make the deployment secure consider each wallet to manage on average 32x the average amount of tNotes owned by wallets, this is constant # as each block the amount of wallets and and tNotes range activated are shared with everyone.
  - tNote does not change ownership on its own, there's a minimum of 17 tNotes transferring together
- **Wallet/Node/Manager/Miner** *// In TodaNote they can be synonymous*
  - While wallets have unique numbers in the Todatree, at the t96 level they can also be part of a branch at a higher level than t96
  - The wallet owning the tNote is the only one able to crypto-sign it to change its ownership
- **tNotes**
  - Each tNote has a unique number at t0 level in Todatree that is atomic, so it is not divisible *// some implementation may suggest level 32*
  - Each tNote is a file, as tNote changes ownership, it keeps a stamp of the location it occupied in its own file, so anyone can compute the wallet chain

it was owned by *// To keep the file small, we would only need the history of the first 3 wallets and last 9 along with the Merkle proof*

- **Decentralized and Distributed Consensus Mechanism**
    - For every tNote, at least 17 managers must testify its transfer for it to be transacted and those managers must be the ones selected by the pseudorandom function so everyone knows *(can compute)* who they should be during a certain block *// Deterministic*
    - Testifying includes the check for authenticity of tNote, ownership, cryptosignature, then racing with that tNote up the Merkle tree for the PoAW
    - The first to deliver to the new tNote owner the merkle proof gets the 0.1% reward in a probabilistic way *// means approximately every 1000 files you testify you get one file*
- **Economical Incentive / PoAW (Proof of Actual Work)** *(Actual work is the distributed computing needed to run the system with maximum replication of 32 average expected 17)*
    - The first validator node / manager for every tNote to get the correct Merkle root will collect the transaction fee of 0.1% *// Getting the merkle root involves building the branch of the Merkle Tree Upwards starting with coupling with other Nodes at L1 all the way to L(root) There are many algorithms that have proofs for al nodes when they received the result of something (here Merkle proof with merkle root) without showing the entire root*
    - Tx fees provenance are split between payer, payee and new issuance via quarks of hashcash like to probabilistically transfer one coin every 3000 times for paying and 2500 for issuance. That’s because initially we are suggesting 0.03%, payee 0.03% and new issuance of tNotes 0.04%. Those numbers seem to make economical sense for microtransactions.
    - While newly issued Tx Notes are in quarks, the fees paid by payer and payer are better sent in a probabilistic formula where every 3000 times an entire tNote of same denomination is sent from payer to validator and from payee to validator
    - Those rates are the minimum to ensure at starting point overall systems security, it isn’t recommended to have any implementation with lower percentages. It is also important to note that the ratio between newly issued notes and paid by sender and receiver must remain the same if an implementation wishes to increase the transaction fees.
    - Transaction fees can automatically be suggested on the node level, so that large amounts are not too costly and economy of nest price will take its course. However 4 important points to be taken in considerations on the

protocol level the ratio matters a lot to prevent from certain malicious attacks and DDOS attacks. :

    - If tx percentages are high then receiver will likely wait for merkle proofs from cheaper nodes first
    - If tx percentages are too low, then validators may not be incentivized and likely not to do the work, so sender and receiver will need to increase the percentage
    - The new quarks generated will always be less than that of the combination of payer and payee combined but more than anyone of them on their own. The suggested for implementation is 40% of the tx percentage total is to be issued in new Quarks. In this situation 0.1% total tx fee is comprised by 0.04% in new Quarks.
    - Some implementation may wish to have an economic disincentive against validating nodes. Where validators would issue a conditional transaction that if they cheat they get penalized and money is taken away from them. This can increase the security but it can also introduce attack vectors so we caution against any disincentive that is higher than the aggregate generated by a node. *//See Wallet reputation section*

- **Todatree structure**
  - T96 level is divided into 4 equal sections that are all on the same level
  - The ones on the far left are for wallets with tNote provenance but the ones on the far right of t96 are for wallets assignments
- **Wallet points/reputation (needed if you wish to implement a hybrid POS with PoAW)**
  - Every wallet owner is automatically assigned a transaction wallet that can only receive tNotes(Quarks) based on transaction fees earned due to PoAW *(Proof of Actual work)*
  - Transaction wallets fill up quarks/tNotes in its branch of the Todatree from left to right. So verifying the one on the very far right you can deduce the reputation as you must leave some Quarks in Tx Wallet at all times, precisely the last ones received indicate the total you received which indicates how many times the node received tx fees for PoAW performed and therefore is compliant with the protocol so you are likely to reach out to them first to validate your transaction, so the more you are compliant the more you are assigned up to a limit of 32X which is the upper limit.
  - Wallet reputation along with other impactful elements on Proof of Stake, like how many tNotes are owned by a wallet and other elements, how long have they owned the tNotes, the diverse provenance of their tNotes and the frequency and disparity of last transactions are some aspects that can play

important roles in impacting the role one wallet can play to participate in PoAW

    - Precisely the algorithm that determines who gets the 0.1% tx fee groups nodes by their combined reputational aspects. So the one with the highest, gets evaluated first. Only if doesn't have the proof, then second group is evaluated, etc. The grouping is by the following elements sorted descending by the value. Transaction fees paid by node in its lifetime, amount of tNotes, Date of last transaction, amount of transaction fees received
    - Even going through all the above, if the only correct Merkle Proof is brought by a node that was just joined the system and is the last to be evaluated, it would still be the one that wins the transaction fee. This prevents certain groups that are at the advantage of being in the system for longer than others to maliciously retrieve a wrong merkle proof. Any new comer to the system could prove them false.
    - Algorithms running the nodes are incentivized to evolve and compete all the time to get the highest rating they are also incentivized to have memory and remember proven bad acting nodes and reduce their impact to the last group to be evaluated. Basically after newly joined nodes.
    - Nodes when validating, can choose to buy trust by having a disincentive to cheat. This basically works in a way that the validating node issues a conditional transaction of certain amount of tNotes (as low as one) if they are proven cheating they lose the entire amount.
    - The math can work in a way that inorder to game the system or initiate a successful sybil attack, you must have full control over more than 50% of the active/issued tNotes in the system AND have paid the most amount of transaction fees, and operate more than 50% of active Wallet
    - Because we don't have any central place to store points/reputation Basically Wallets reputation can be stored with the Wallet itself as a series of merkle proofs along with ascending markers using Toda Tree structure.

- **Transaction stamp / Double Spending**
    - Payer and payee must cryptographically sign inorder for Units to be transacted
    - Up to 32 from the pseudorandomly selected Nodes/Wallets by a function of the last Merkleroot so it's deterministic and everyone knows who they should be for any active tNote for the duration of a certain block. If a tNote is transacted, they will all be replaced by a new set of 32
    - The first of the 32 selected nodes that comes back with the correct merkle proof (including new merkle root) will get the 0.1% transaction fee - We call this PoAW // *proof of actual work, the job required to bring back the merkle proof is described in section PoAW*

- TODAQ or Quarks, are managed in a value as an extension to the tNote parent. Given that users can not send Quarks on their own but as part of transaction fee they can only receive Quarks or their Quarks gets subtracted when they issue a transaction.
- Although tNote authenticity can be checked up instantaneously, in-order for tNote receiver to ensure every tNote received was not double spent, it must wait for the block to conclude and stamp transaction with Merkle root and retain Merkle proof for use in subsequent transaction. At the end of the block, the tNote transacted will have new managers that are pseudo-randomly chosen by a function of the merkle root, those new managers must certify ownership to only one owner and therefore anyone in the system including new owner can easily check to see if tNotes are assigned to them and no one else.

Please Note: This is a summary of the core protocol with all the components needed for the protocol. For references, credits, appendixes, details please contact the authors. For implementation details, depends on which implementation and if you are authorized we would always be happy to share all our work and that of the community without infringing on any trade secret and NDAs which is why we share with caution.
This summary was written by co-author Toufi Saliba and reviewed by co-author Dann Toliver and over 30 crypto contributors and reviewers. It's open to re-use under CC BY-SA 4.0