TODA Protocol - A Ledgerless Blockchain

A new approach for a Maximum Distributed and Decentralized Protocol

Intro:

There is currently a need for a strong governance *(decentralized)* value exchange platform that uses maximum distributed computing so it is efficient AND scales to millions of transactions per second, powering billions of devices while maximizing security and minimizing transaction fees *(to almost zero)*. The TODA protocol is designed to meet those needs, is fully distributed (BFT, *efficient yet redundancy x32, reliable and secure)*, has a governance model which becomes more and more decentralized as it grows, and is optimized for mobile devices without computer desktops or server or 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 in early releases, 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, Machines 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 Machines that it could potentially interact with. We use it because it is extremely fast to compute at each Machines' level to locate a certain leafs' 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.*

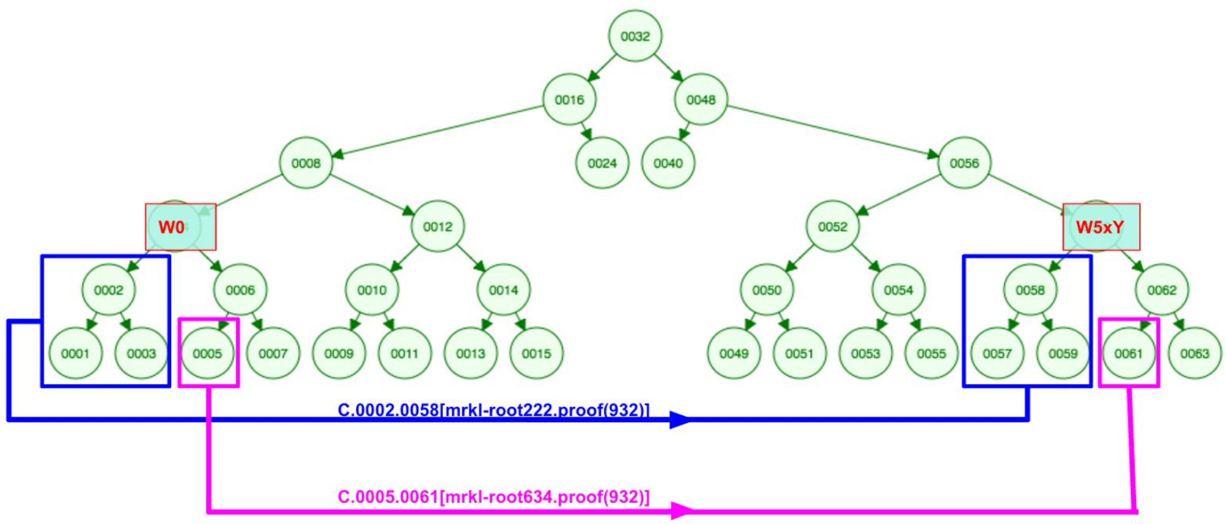
The smallest Unit is at the leaf's level we call it TODAQ or Quark. However the smallest unit that can not be divided by users but it can be transferred in its wholly by users we refer to its structure as atomic and we call it TodaNote, tNote, TODAUnit or even Toda File because it is actually a file. It has an atomic structure, it has a unique number through its life and the quarks are necessary to 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.

// This abstraction is similar to atoms 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 Machine *(technically 2^63 tNotes is the maximum limit to be initiated into the system hint: 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 Machines 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 Machine 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 Machine, 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 malicious attacks and reduces the success rate 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 0.04% of new quarks that are newly created / issued from provenance every transaction that can be sent to the transaction Machine of users who win the PoAW. 2^32 quarks = one tNote.
    - The Quarks are actually the leafs of the branch TodaNote in todatree*
    - Implementation recommendation would be using a hashcash like of next block's merkleroot concat of existing transaction hash to charge 1 entire tNote only 0.03% of the times which effectively be the same results but don't have to deal with splitting off the tNote at the user level. The Toda Protocol CORE Implementation uses this method. *// We do acknowledge that this method is not as cashflow efficient as you have to at least 1 TodaNote per 3000 TodaNote transacted. Basically, because it is 0.04% of new tNotes are created per tNote movement, a single 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 rather than the full tree, but we keep 256 by design for ease of future expansion, so each implementation can take on a branch of level 160, *//technically speaking you can have 2^96 branches at level 160 and therefore many implementations can benefit from interoperability*

Todatree simplified 256 to 5



This tree is used as a map so Machines 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 call this tree the Todatree. We refer to level 96 as t96 and it is the level that represents the Machines/Machines. *// some implementations suggestions Machines are at 64*

- The very first 160 Machines on the left of Todatree level 92, have the denominations and are setup in a way where tNotes provenance of all the tNotes *// other branches can have units for smart contracts or whatnot*
- Quarks are at the todatree leafs level t0 of that Machine, However not all quarks have same value, they all depend on the provenance Machine, originally we recommend denominations to be represented in left or right Machines 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 Denomination makes it 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 Machines, hence Machine chain is within each and every tNote is calculated at the system level as it is better from usability perspective to show Machine chain than point occupied chain, especially if your Machine remembers other Machines and can give them names. Kinda like 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 Machine 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 Machine = (W(2^95 + (block# * (2^96))) - *//This is important to prevent unauthorized tNote creation/activation*
- **Merkle Tree** //see appendix
    - Generated every T seconds
    - The root hash is generated by every active validating Machine in the network shared across the network *// You absolutely need 1/32 or about 3% of the network to be active for merkle tree to be built*
    - 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 the Machines/managers of those tNotes (group A) along with the merkle root prior (Group B). Each merkle root when inserted in the pseudorandom function can assign 32 managers. Hence a total of 64 managers (groupA + groupB) **can** work on building the merkle proof. *// Note "can" but don't have to*
    - Coupling of tNotes is deterministic from Toda Tree reference pointers. If tNote unique number X is even then it couples with X+1 otherwise X-1
    - The pseudorandom function using the tNote number can give the Managers Toda Note addresses
    - If a manager from Group A does not have merkle proof in subsequent block when it moves to Group B, then manager can not win the 0.1% even if manager generates the merkle proof then. Therefore Managers must have it before and build it again with new values when tNote is being transacted, so it is likely that only up to 32 are incentivized to build the merkle proof at any given time. *// If for example Managers of tNote22 build the merkle proof in block 333 but then in block 334 tNote22 isn't transacted, they have no incentive and won;t likely build it*

- Basically every Machine 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 during any specific block
    - Dormant or disconnected Machines, when they rejoin the network using gossip like protocol can get the update of last list of merkle roots
    - Every Machine is expected to have all Merkle roots in the system along with last tNote assigned and last Machine assigned during each Merkle root *// that's basically the only universally shared data if a Machine 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 Machine which tNotes to manage during the existing block
        - To make the deployment secure consider each Machine to manage on average 32x the average amount of tNotes owned by Machines, this is constant # as each block the amount of Machines 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/Machine/Manager/Miner/Node** *// In TodaNote they can be synonymous*
    - While Machines 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 Machine 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 Machine chain it was owned by *// To keep the file small, we would only need the history of the first 3 Machines and last 9 along with the Merkle proof*
- **Decentralized and Distributed Consensus Mechanism**
    - For every tNote, at least 1 of 32 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 *// The selection is 32 if many provide different*

data, as long as 17 is providing the same, the Note gets accepted by receiver and transfer is complete.

- Testifying includes the check for authenticity of tNote, ownership, cryptosignature
- And PoAW (Proof of Actual Work) is basically racing with that tNote up the Merkle tree in that block and obtaining the merkle proof and delivering to payee.
- The first to deliver to the new tNote owner the merkle proof gets the 0.1% reward in a probabilistic way *// approximately every 1000 files you testify during exchange and deliver the merkleproof first, you get one file/tNote*

- **Economical Incentive / PoAW (Proof of Actual Work)** *(Actual work is the distributed computing needed to run the system with maximum replication of 32)*
    - The first validator Machine / 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 Machines at L1 all the way to L(root) There are many algorithms that have proofs for all Machines 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 payer 0.03%, payee 0.03% and new issuance of tNotes 0.04%. Those numbers seem to make economical sense for microtransactions.*// Effectively the cost of transaction fees is approximately the aggregate cost of network, electricity and depreciation. Basically the system runs itself and pays for itself. However, certain implementations may wish to have different percentages. Higher or lower may increase certain risks of successful attack vectors.*
    - While newly issued Tx Notes are in quarks, the fees paid by payer and payee 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
- **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 Machines with tNote provenance but the ones on the far right of t96 are for Machines assignments
- **Transaction stamp / Double Spending**
    - Payer and payee must cryptographically sign in-order for Notes to be transacted

- Up to 32 from the pseudorandomly selected Machines/Machines 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 Machines 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*
- 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 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. For references, credits, appendixes, details please contact the authors. For implementation details, depends on which implementation we would be happy to share all our work and that of the community without infringing on any trade secret and NDAs. This summary was written by co-author Toufi Saliba and reviewed by co-author Dann Toliver of Toda Protocol that will also be credited to over 30 crypto contributors and reviewers. Full list of credits will be made available. Although some components are either patented or in patent process, the Toda Protocol is open to re-use under CC BY-SA 4.0