# TODA Protocol: Managing and Transmitting Value at the Network Packet Level

An Ongoing Quest to Security (Decentralization BFT), Efficiency, Confidentiality, Scalability and Interoperability

By Toufi Saliba and Dann Toliver

Last edited: October 2018

**Abstract.** We describe a value management and exchange protocol on the network packet layer that offers a significant increase in security (maximum decentralization) and decrease in the asymptotic complexity net cost per transaction (over 99%) when compared to any current system that either depend on centralized or decentralized system that are often third parties to the users transacting. This allows transactions of pseudo-randomly selected nodes (up to 32 redundancy) for each packet along with self-validating atomic units to be performed with a total energy usage in the system that is a log factor of the number of users; Hence given a consistent average number of transactions per second per user the total number of transactions per second is essentially unbounded. The Toda protocol security, efficiency and scalability increase as the number of devices increase. As long as active devices are geographically distributed, they can contribute to the global governance of the system by design and measured by the speed of network (almost the speed of light which is a universal constant that can not be exceeded)

**The intent of Toda is to enable economies with self sustainability without any extraction of wealth or dependency on external economies resulting in an ongoing user-centric value added**

*Important: Toda Protocol is not a ledger based system and doesn't depend on ledger. It is designed to eliminate anyone in the middle for any value management system to exchange between each other so no dependency on exchanges either. Inorder for systems to be interoperable on a P2P basis while benefiting from efficient unbounded scalability due to security first by design. Toda is not "another blockchain" Toda is based on a modification of network packets so they carry value and change ownership impacting a single globally replicated hash every block of time. For those familiar with Bitcoin, an implementation of Bitcoin On Toda instead of current TCP/IP treats each Satoshi as a packet which can also make it P2P exchangeable with implementations of Zcash, Ethereum, EOS and any other blockchain On Toda.*

**Note**

This V2+ whitepaper summary is private, has some information of several implementations of Toda. It may contain confidential information, and is not intended to be shared or released publicly. Every implementation code, will be open sourced prior to its release. The protocol as described herein is accurate, but there are deliberate gaps in the details especially on algorithms used to perform certain function until it's fully open source.

## 1. Exordium

The reduction of cost for management and transmission of value on the network layer in a decentralized BFT setting can present opportunities to businesses and individual to implement systems that can be self sustainable. Inorder to do that such network must meet all the following 5 necessities: Security, efficiency, confidentiality, scalability and interoperability.

There are 2 types of values. The first a single entity care about, The second kind, more than one entity care about.

- The first one doesn't need to be secured, nor transmitted.
- The second one must be secured and often transmitted (change ownership).

In-order to secure at rest and during transmission digitally, many established "trusted" organizations (like government agencies for car registry or banks for money) they end up with a lot of friction because that control they have will put them in a position to follow lengthy policy and procedure and get audited and examined by other third parties who would also need to be governed by other groups and so on.

Those established organizations built ledgers to query which then was revolutionize with Bitcoin to make replicated ledgers across many machines and a consensus mechanism to ensure each device is representing a single device and therefore strong immunity to voting manipulations etc.

However, inherently a replicated ledger has a lot of frictions even if the consensus mechanism is solved.

A new approach on the network layer has been extensively and successfully tested using deterministic distributed computing (DDC) as the "work" in the proof of work (PoW), here the DDC is what is needed to run the system itself while the "proof" in the PoW is hash of deterministically constructed merkle tree using a fully saturated Binary Search Tree structure as a map to contribute into structuring the temporal coupling for each and every representation. Which is the hashes of all hashes to get to the Merkle Root of the current block that is being constructed.

The evolution of TODA Protocol continues towards the next evolution of TCP/IP (not replacing TCP/IP). This can help every current and more so future blockchain technology and every technology intending to transmit value to get additional security, scale efficiently and enable exchanges between blockchains without depending on any third party.

### Opportunity TODA-T as an International Open Standard

We propose a standard that has unique number for each and every packet (similar to how IPV6 gives unique number for each and every device, except here it is in a fully decentralized setting, but it would give unique numbers for all objects in the entire network, including the smallest files/packets that can represent an asset or value which can be transacted utilizing the Toda-Protocol. At the OS level they are viewed as files, however at the network communication layer with V2+ they are viewed as network packets and can run

below the OS.

As a primary consideration, every attempt has been made to limit the use of expensive asymmetric signing and verification requirements in favour of efficient secure hashing techniques. Indeed, even early prototypes have been successfully tested and continue to be developed on mid-market consumer mobile devices - a hardware platform unthinkable for earlier generation currency designs.

Pseudorandomly yet deterministically, a set of active network devices geographically dispersed are selected during each block of time for each packet. In certain implementations only when packets are being transmitted/change ownership the selected devices are activated, however, in V2.05+ we describe a system that will be incentivized through every block of time to take on the packets or group (branch from Toda-Tree) of packets unique ID from Toda-Tree though the Toda-Merkle-Tree. On average, if each and every device has abou 8 packets to do the work, it would result in minimal work that is essential to be part of the network. If for example Bitcoin is implemented on Toda-T instead of TCP/IP, the 20+ millions active users will no longer depend on replicated ledger but instead on much larger set as all users become part of the system in much smaller replication (up to 32 per packet), effectively the mining rewards would be distributed to all active users, at the time this has been calculated, an average of 0.003 Bitcoin per user per year is more than what would the network cost on average. However, the throughput is estimated to increase to 24,000 transactions per second due to security-first by design (full decentralization) and distributed computing. Furthermore, the interoperability between blockchains will become truly peer-2-peer and not dependant on any third party.

The current problems blockchains face when implemented on TCP/IP is that they must depend on global replicated ledger which is costly by design to propagate. We must not forget what got us here are repeated exploitations of the Bitcoin Protocol that lowered the throughput expectations of every blockchain. Bitcoin by design was intended to deliver on what is the ongoing quest of Toda Protocol. Although currently out of all blockchains, we are only helping Bitcoin due to bandwidth, the success of scaling at this layer will benefit every single blockchain technology that was built and needs to be ported or will be built, but only when developers they realize Toda's potential[1].

## 2. File/Packet structure

Each atomic unit of value has a kernel, which is structured as a record with the following properties:

`payload`: identifier content (arbitrary), 2^12 bytes maximum

`creation cycle`: current global Merkle root

`initial_wallet_identifier`: the wallet identifier of initial account

`signature`: the signature of the initial wallet

The unit's ID is the hash of the kernel. In this document unit is often referred to as a 'file', 'packet' or a 'note', and occasionally as a 'coin'. Effectively it is a Toda-T Network Packet.

---

[1]**Sources:**
https://en.wikipedia.org/wiki/Circular_economy https://www.ellenmacarthurfoundation.org/circular-economy
https://digiconomist.net/ethereum-energy-consumption
https://cointelegraph.com/news/ethereum-mining-needs-more-energy-than-cyprus-cambodia -brunei

Each transaction causes a transaction block to be appended to the packet. Each transaction block has the following structure:

`cycle:` the global merkle root for this transaction

`previous_hash:` the hash of the packet before this tx block was added

`sender_wallet_identifier:` the wallet identifier of the sender

`sender_signature:` the signature of the sender of the packet

`recipient_wallet_identifier:` the wallet identifier of the recipient

`recipient_signature:` the signature of the recipient of the packet

`validator_signatures:` an array of signatures from the appropriate validators

### 3. Transaction processing

Given a packet F owned by wallet A, that packet can be sent to wallet B through the generation of a transaction request, which is first signed by wallet A, then signed by wallet B, then distributed and signed by a set of validators (any active device that is geographically dispersed is a validator in every block to some packets) who are deterministically chosen for this particular transaction within this particular block of time. For a more thorough account of the transaction processing protocol please see the transaction protocol specification below.

### 4. Validation

For each transaction, a set of validators is deterministically chosen based on the packet identifier (the hash of the kernel) and the current merkle root. This provides two important qualities: a fixed amount of work is done per transaction, regardless of the size of the system; and that work is spread out through the system, rather than concentrated as it would be in a replicated ledger. Each validator provides four important functions:

1. Confirming the validity of the transaction (structural soundness and proof correctness)
2. Prevent sending a packet twice in this cycle
3. Help build the consensus proofs for the transaction
4. Provide matching proofs for A and B

### 5. Consensus

Within each block a merkle tree is built (not by a single device but all devices in the network combined) which contains proofs for each transaction. For a given transaction Tx its two proofs

(one from the sender, one from the receiver) will be known only to the sender, the receiver, and their validators. A binary merkle tree is built using a deterministic selection algorithm to choose a partner. The pseudorandom function uses the current block hash aka the merkle root. Once selection is determined, the temporal coupling will take place creating a hash out of both packets, we call that Level 1 in the Merkle Tree, then constructing Level 2 would be of the hashes of concat hashes of Level 1, and so on. (In Satoshi.OT - an implementation of Bitcoin On Toda-T - only 51 levels are needed so each and every Satoshi is represented by a unique packet).

For more information on the TODA trees and the consensus-building process please see the TODA Technical paper.

## 6. New Packet/Note creation

In each transaction, the sender and receiver send a conditional transaction request to each of the 32 validators for each packet, this allows that validator to start constructing the merkle tree. Additionally, each validator has an even smaller chance of creating a new Toda Note based on the next merkle root. The payload of the new note is the signed transaction validation message the validator returns to the recipient, which must match the new merkle root to the first k bits. When this match occurs, the validator is allowed to assign this new note to themselves on the block/cycle immediately after the one in which the transaction took place, by performing a transaction of the new note to themselves (with appropriate validator signatures, proofs, etc).

Depending on the implementation, in Bitcoin for example, Satoshi.OT would be a user activated fork if adopted by current Bitcoin like BTC and/or BCH. Another example would be Ethereum and in that case maybe full on new deployment or existing forks of ETH and/or ETC. A successful one we are also aware of is the implementation at TodaQ as mentioned earlier. A set of Toda Notes backed by a trust structure consisting of existing fiat currency are created during the initial genesis block. These packets have a special structure to allow their creation to be independently confirmed, which is important because unlike a ledger-based system the TODA protocol does not provide a mechanism for querying which packets belong to which accounts or how much an account contains.

During initial note generation, each unit’s payload is a consecutive integer, which functions like a serial number. These notes are assigned to a known set of wallets, which sign for them during the next block/cycle, as in the new note creation mechanism. The genesis block can then be regenerated from scratch by any third party, ensuring the integrity of the initial note generation.

# TODA Protocol Technical Summary

**Please Note:** This summary was written by co-author Toufi Saliba and reviewed by co-author Dann Toliver of Toda Protocol. It will also be credited to dozen crypto contributors and reviewers. Full list of credits will be made available. Although some components are either patented or patent pending to not restrict any of its utilization, the Toda Protocol is open to re-use under CC BY-SA 4.0

*At the time this protocol is being written, an entire industry of Miners is emerging and will continue to fight every crypto that doesn't require mining. Toda can enable any existing blockchain to scale, miners can provide value as in routers, accelerators etc. Also, Financial services to provide "true services" Toda will only encourage those implementations as long as users don't depend on them and can run without them. Basically governance must always be with users. Ask authors about some development in Biometrics and Proof Of Walk, if interested in helping out there as well.*

**Introduction**

Most internet users have smartphones and can't afford mining machines. There is currently a need for a single class, secure strong governance *(decentralized)* value exchange that uses maximum distributed computing so it is efficient AND scales to millions of transactions per second when powered/powering billions of devices *(single class of machines)* while maximizing security and minimizing transaction fees *(to almost zero)*. The TODA protocol is geared to meet those needs:, it is fully distributed *(BFT, efficient yet redundancy x32)*, has a governance model which becomes increasingly more and more decentralized as it grows, and it is optimized for mobile devices without computer desktops or server or the need for any centrally controlled system. It can also scale existing blockchains if they use its rails wherebyand nodes can optionally run on a combination of mobile devices and nano-micro cloud instances, each one, one representing a unique each mobile device. *// This step might be necessary in early releases to increase social scalability, so it's highly recommended that implementation details take that into effect. See PoDW for details of how "more than necessary" cloud power does not make any difference, folks refer to that as CMR (Centralized Mining Resistant)*

**Design overview**

In a BFT decentralized system that uses distributed computing, Machines will need a reliable 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 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 to them and are empty when they aren't occupied by them. 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 simplified to any computer scientist to comprehend it and imagine its navigation and the possibilities of things that can be done deterministically. y*// This map/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 using DAGs, DHT CHORD or other.*

The smallest Unit is at the leaf's level we call it 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 Toda-T Network Packet, Note or Toda File because it is actually a file when viewed by Machine's OS and when implementation below OS it is viewed as Network Packet. Depends on the implementation, with V1 only files are used. 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 Toda Packet and can not form another Toda Packet. Quark numbers are like DNA they belong to one single Toda Packet and can not belong to 2 Toda Packets at the same time. *// 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 unless you are working on a nuclear reaction. So one set of 2^32 quarks can form one specific Toda Note with one unique number and not any other Toda Note*

One application of TODA protocol, a value to be exchanged similar to a currency, and the provenance of each unit/Toda Note is from a sub-branch at level 96 the Provenance Virtual Branch *(technically 2^63 Toda Notes is the maximum limit to be initiated into the system hint: Toda Notes are at level 32)* - A Toda Note can travel within the Todatree when cryptographically signed by owner to another, and have a new location every block only when transacted. Every movement builds a chain of Machines/Wallets edited inside the Toda Packet *(also at level 96 in todatree, some implementation details may suggest level 64 which aren't concerned with the quarks, in those systems they either don't care about the ongoing supply/creation of Toda Notes or have them in a probabilistic function that results to similar outcome).*

- **Atomic Toda Notes** *// transaction fees and creation of new Toda Notes*
    - Each unit/Toda Packet contains its own history as a Machine chain of past transactions. If UserA sends UserB Toda Packet C115, from the user's perspective, the Toda Packet would

look like this: C115_W(UserA).BlockNumber1111.MerkleProof1111.7854_to_W(UserB) .BlockNumber88888.MerkleProof88888.569 *// To the machine, each Toda Packet can only occupy certain location in the Machine which is a branch and therefore has a unique number derived from TodaTree in the entire system.*

○ Toda Note/Packet/File is the smallest unit that can be transacted by users *// In early implementations we are recommending the mininum suggested Toda Notes to be transacted to be 17, this increases the cost of malicious attacks and reduces the success rate significantly that way Toda Notes include the merkle proof of theirs and that of 16 others, using denominations almost any transaction can be expressed with up to 23 Toda Notes.*

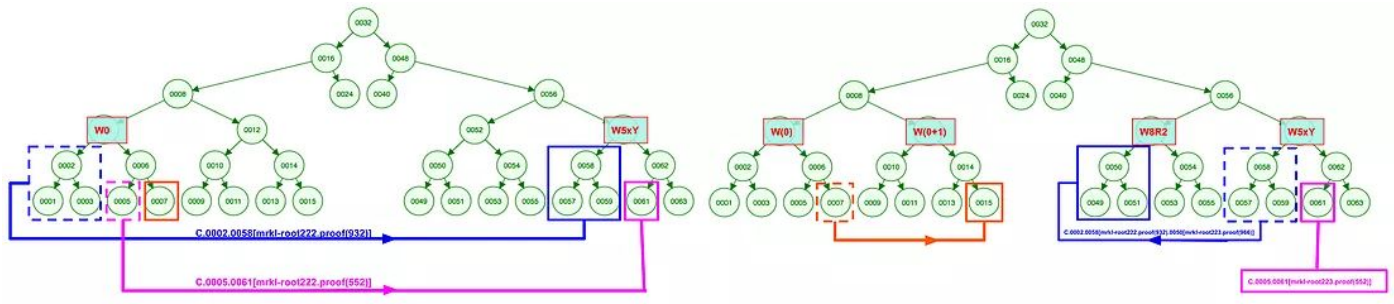
○ Quark is the smallest unit non-divisible, it is in some of non-probabilistic implementations used for 0.04% of new quarks that are newly created / issued from provenance branch of TodaTree every transaction that can be sent to the transaction Machine of users who win the PoDW. 2^32 quarks = one Toda Note/Packet.

○ The Quarks are actually the leafs of the branch Toda Note/Packet in todatree*

○ Implementation recommendation would be using a hashcash like of next block's merkleroot concat of existing transaction hash to charge 1 entire Toda Note only 0.03% of the times which effectively be the same results but don't have to deal with splitting off the Toda Note/Packet at the user level. The Toda Protocol CORE Implementation uses this method. *// This method is not the most cashflow efficient as payer and payee must put 1 Toda Note per ~3333 Toda Note transacted. Also, because it is 0.04% of new Toda Notes are created per Toda Note movement, a single Toda Note will have to change ownership 2500 times so its impact aggregate is creating 1 new Toda Note*

· **Map/Todatree**

o A BST data structure 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 data structure / map so Machines know how to navigate the system and where to expect objects to be. *// Think of IPV6 but fully decentralized. 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 *// Some previous implementations suggested Machines are at 64 without Quarks*

- T96 level is where all Machines are at. *// bitcoin on Toda would be a branch level 83 because 2^51 = all the satoshis to ever exist per Bitcoin implementation of 21 million bitcoin max.*
- A higher level at T160 can be an implementation all on its own. It is wize for implementations to choose a unique T160 so they can be exchanged with other implementations without depending on any intermediary. *// There are 2^96 to choose from*
- The ones on the far left are for Machines with Toda Note provenance but the ones on the far right of t96 are for Machines assignments like Wallets/Users/Smartphone.
- The very first 160 Machines on the left of Todatree level 96, have the denominations and are setup in a way where Toda Notes provenance of all the Toda Notes *// 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 Branch of TodaTree, each implementation can have different branches for different denominations and different values. Initially we recommend denominations to be represented in left or right Machines such as W100 is where the Toda Notes, 20 to the left would 0.01 TODA 20 to the right would be 100 TODAs, 30 would 1000 TODAs Denomination makes it easier to send small or large amounts, can be computed in a combination that will always be minimum 17 Toda Notes during one transaction. The average amount of packets per transaction will depend on the economy using it, but will likely be between 20 and 30.

· Toda Notes 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 Toda Note is calculated at the system level as it is better from usability perspective to show Machine *(wallet chain / user chain)* 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 Toda Note number you can tell its provenance block by conducting a quick binary computation, so the provenance W of a Toda Note 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 Toda Note creation/activation*

· **Merkle Tree** *// for more dev specs see appendix 13*

o Generated every T seconds *// T can be 30 seconds initially but it is adjusted dynamically every X blocks, intent is to have majority of transactions requesting change of Toda Note ownership make it in the block they request, so average time for capability to re-spend is less than 2T*

o The merkle root (root hash) is generated by every active validating Machine in the network shared across the network *// You only need 1/32 or about 3% of the network to be active for merkle tree to be built every block if there are no transactions and you would need over 50% if there are transactions.*

o Every Toda Packet is coupled with another Toda Packet to potentially form L1 (L1 for Level one) of new merkle tree *// Merkle tree and Todatree are not the same, merkle tree gets created every block based on the provenance branch, basically the provenance branch, gets replicated every block with only up to the Toda Packets issued while those in branch not issued will always have static value until issued, in Bitcoin On Toda for example the provenance branch is 2^51 / however the packets can be at much higher level until there are many more users, so each packet can represent 1024 Satoshis.*

o Utilizing the previous Merkle root in a function to pseudorandomly select the Machines/managers of those Toda Packets (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. *//"can" but don't have to*

o Coupling of Toda Packets is deterministic from Toda Tree reference pointers. If Toda Packet unique number X is even then it couples with X+1 otherwise X-1

o The pseudorandom function using the Toda Packet number can give the Managers Toda Packet addresses

o 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 have an incentive to have before in anticipation of building it again with new values when Toda Packet is being transacted *// If for example Managers of Toda Packet22 build the merkle proof in block 333 but then in block 334 Toda Packet22 isn't transacted, they have no incentive and won't likely build it*

o Basically every Machine when acting as manager is able to run the pseudorandom function that provides the coupling for each and everyone of its Toda Packets that it is managing during any specific block.

o Dormant or disconnected Machines, when they rejoin the network using gossip like protocol can get the update of last list of merkle roots

o Every Machine when delivering the merkle proof can not win the 0.1% if it doesn't have the previous one *(therefore was online)* and if it didn't pass the entire data it received to Group B. This creates an ongoing incentive for Machines to receive and deliver on all the Toda Notes despite the value. This is important, without this, Machines are only incentivized to handle high denominations, and high value.

o Every Machine is expected to have all Merkle roots in the system along with last Toda Packet 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**

o 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 Toda Packets to manage during the existing block

§ To make the deployment secure consider each Machine to manage on average 32x the average amount of Toda Packets owned by Machines, this is constant # as each block the amount of Machines and and Toda Packets range activated are shared with everyone.

o Toda Packet does not change ownership on its own, there's a minimum of 17 Toda Packets transferring together

**· Wallet/Machine/Manager/Miner/Node** *// In Toda they can be synonymous*

o 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 o The Machine owning the Toda Packet is the only one able to crypto-sign it to change its ownership

**· Toda Notes (Packets, Files Are used interchangeably)**

o Each Toda Packet has a unique number at t0 level in Todatree that is atomic, so it is not divisible *// some implementation may suggest level 32*

o Each Toda Note is a packet or a file, as Toda Note changes ownership, it keeps a stamp of the location it occupied in its own packet/file, so anyone can compute the Machine chain it was owned by *// If to keep the packet small, we would only need the history of the first 3 Machines and last 9 along with the Merkle proof, some implementations may want to offset some of the history somewhere else, like a backup or transaction log*

**· Decentralized and Distributed Consensus Mechanism**

o For every Toda Packet, 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 are providing the same, the Packet gets accepted by receiver and transfer is complete. Given that transaction must be a minimum of 17 Toda Packet, receiver can be satisfied with less than 17 per Toda Packets because information of transaction is propagated in the other packets.*

o Testifying includes the check for authenticity of ownership proofs & signatures

o PoDW (Proof of Distributed Work) is basically racing with that Toda Packet up the Merkle tree in that block and obtaining the merkle proof and delivering to payee.

o The first to deliver to the new Toda Packet owner the merkle proof gets the 0.1% reward in a probabilistic way *// Some implementations may choose pseudorandom function of next mrklroot to determine the receiver of 0.1% rather than the first. Approximately every 1000 packets you testify during exchange and deliver the merkleproof first, you get one packet/Toda Packet*

· **Economical Incentive and DDOS prevention / PoDW (Proof of Distributed Work)** *//Proof of Distributed Work is the distributed computing needed to run the system with replication of 32x*

o The first validator Machine / manager for every Toda Packet to get the correct Merkle root will collect the transaction fee *// Getting the merkle root involves building the branch of the Merkle Tree in a distributed way, starting with coupling with other Machines at L1 all the way to L(root) .. Please Note: Some implementation may choose to give not to "first" but based on a pseudorandom function of next block's hash / mrklroot. Each implementation has its pros/cons.*

o Tx fees are split between payer, payee and new issuance via quarks of hashcash with low probabilistic outcome to transfer one coin for paying and for issuance. The protocol will suggest fee algorithms for payer, payee and new issuance of Toda Packets that make economic sense for micro-transactions. // Effectively the cost of tx is approximately the aggregate cost of network, electricity and depreciation, implementation can set it to adjust automatically by bidding the lowest fees, if your bid is too high or your network or machine too slow you'll less likely receive the 0.1%. Basically the system runs itself and pays for itself. However, certain implementations may wish to have different fix percentages. Higher or lower may increase certain risks of successful attack vectors. While newly issued Tx Notes are in quarks, the additional fees paid by payer and payee are easier if sent in a probabilistic formula where every ~3333 times an entire Toda Note of same denomination is sent from payer to validator and from payee to validator

· **Transaction stamp / Double Spending** *// For more dev specs see appendix 13 and 14*

○ Payer and payee must cryptographically sign in-order for Packets to be transacted (yes receiver pays its network in Toda)

○ Up to 32 from the pseudorandomly selected Machines/Machines by a function of the last Merkle root so it's deterministic and everyone knows who they should be for any active

Toda Packet for the duration of a certain block. If a Toda Packet 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 *// PoDW*

○ Quarks, are managed in a value as an extension to the Toda Packet parent. Given that users can not send Quarks on their own but as part of transaction fee they can only receive Quarks.

○ Although Toda Packet authenticity can be checked up instantaneously, in-order for Toda Packet receiver to ensure every Toda Packet 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 Toda Packet 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 Toda Packets are assigned to them and no one else.

# TODA TRANSACTION PROTOCOL SPECIFICATION

### 1A. TREQ: Transaction Request

**Fields**

- Transaction
  - Sender Waller Identifier
  - Recipient Wallet Identifier
  - Coin Identifier
  - Coin State
  - Sender signature
- Coin file
- Merkle Proof of Provenance
- Current Merkle Root
- Optional Conditional Merkle Root prefix

**Origin**

This message must only be created by clients which believe they hold a valid coin to transact.

**Routing**

This message must only be routed to the client representing the destination wallet, if online.

This message may be held by the originator or an intermediary for some time until the destination client becomes online.

**Receipt**

This message must only be received by the destination client. Recipients who do not match the destination must ignore this message.

**Processing**

The recipient may decline the request with a TFAIL message. The recipient must determine the validity of the request using the check_validity function. If recipient determines a problem, they should publish a TFAIL message and a TERR message. Clients may choose not to publish TFAIL message if they believe they are under a DDoS scenario. See notes in TERR for details. If the recipient validates the request, they should publish TVREQs for each validator reported by the coin_validators function.

If the recipient received insufficient TVALID message in response to the TVREQs, they should issue a TFAIL of type INSUFFICIENT_VALIDATORS, but they may instead choose to DECLINE. If they gather sufficient TVALID messages, they should issue a TWIN. They should also complete the 'processing' steps of TWIN themselves (as if they received the TWIN).

### 1B. TVREQ: Transaction Validation Request

**Fields**

- Validator Wallet Identifier
- TREQ
- TREQ recipient's signature on the Transaction
- Reward (conditional) TREQ

**Origin**

This message must only be created by clients which believe they hold a valid TREQ destined for them.

**Routing**

This message must only be routed to the client representing the validator wallet. This message must not be held by the originator or an intermediary for some time until the destination client becomes online.

**Receipt**

This message must only be received by the validator wallet specified in the request. Recipients who do not match the destination must ignore this message.

### Processing

Clients must verify they are included as a valid validator for this coin using the textttcoin_validators function.

## 1C. TERR: Transaction Error

### Fields

- TREQ or TVREQ
  - – NOT_A_VALIDATOR: This client does not believe they are a valid validator of the Transaction.
  - – INVALID_SENDER_SIG: This client does not believe the coin sender's signature on the Transaction is valid.
  - – INVALID_RECIPIENT_SIG: This client does not believe the coin recipient's signature on the Transaction is valid.
  - – INVALID_COIN: This client does not believe this is a valid coin, at all.
  - – INVALID_PROOF: This client can supply an alternative proof of where this coin should be. – SPENT: This coin has been spent this cycle.
- Optional: Proof of error
- Proof of Valid Validator (POVV)

### Origin

This message must only be created by clients which received a TVREQ where they were listed as the validator, and for which they believe they have encountered a validation error.

### Routing

This message must be routed to the client representing the sender of a TVREQ received by this node. This message must be routed to the sender and receiver of the TREQ. This message must be routed to the other validators of the TREQ. This message may be routed to other clients for the purpose of aware- ness. This message may be held by the originator or an intermediary for some time until the destination client becomes online..

### Receipt

This message must be received by the TVREQ originator wallet specified in this message. This message may be received by other validators of the TREQ while they are attempting to validate the transaction. Recipients who do not match the destination may use the contents of this message to contribute to network awareness and exert prejudice against nodes making provably false requests.

**Processing Notes/Packets**

Recipients must verify the POVV using the valid_povv? function. This message is only generated when the receiver or the sender (or both) are not following the protocol correctly, and that fact is verifiable by clients outside of the transaction.

### 1D. TVALID: Transaction Validation

**Fields**

- TVREQ Originator
- TREQ Identifier
- Validator's signature on TREQ
- Proof of Valid Validator (POVV)

**Origin**

This message must only be created by clients which received valid TVREQ destined for them, and which they believe represents a valid transaction.

**Routing**

This message must only be routed to the client representing the sender of a TVREQ received by this node. This message may be held by the originator or an intermediary for some time until the destination client becomes online.

**Receipt**

This message must be received by the TVREQ originator specified in the request. Recipients who do not match the destination may use the contents of this message to contribute to network awareness.

**Processing**

Recipients must verify the POVV using the valid_povv? function.

### 1E. TFAIL: Transaction Failure

**Fields**

- TREQ Sender
- TREQ Identifier
- Error code:
  - INSUFFICIENT_VALIDATORS: This client received insufficient TVALID messages for the TREQ within the cycle.

– VALIDATION_FAIL: Validators negated this transaction.

– DECLINED: The recipient has declined this transaction.

• Optional: Alternate proof

**Origin**

This message must only be created by clients which received a TREQ where they were listed as the recipient, and for which they believe they have encountered a validation error, or whose validators encountered validation errors.

**Routing**

This message must only be routed to the client representing the sender of a TREQ. This message may be held by the originator or an intermediary for some time until the destination client becomes online.

**Receipt**

This message must only be received by the TREQ originator specified in this message. Recipients who do not match the destination must ignore this message.

**Processing**

Note failure of the transaction.

### 1F. TWIN: Transaction Success

**Fields**

• Coin File

• Accepted Transaction

• List of Validators signatures on the above

• Recipient's signature on the above

**Origin**

This message must only be created by the recipient of a TREQ which has successfully validated the Transaction.

**Routing**

This message must be routed to sender of the associated TREQ, and the validators of that TREQ.

**Receipt**

This message must be received by the sender of the TREQ and its validators. It may also be received by other network participants for awareness, or meta-wallet reporting.

### Processing

Incorporate this transaction success into this block/cycle using the submit transaction function.

## 2. Transaction Processing Functions

submit_transaction incorporates the transaction block into the merkle tree and returns the new coin_file.

```
submit_transaction(transaction, coin_file):
      coin_file.add_tx_block(transaction) new_root = mt.submit_coin_file(coin_file)
      coin_file.add_mpop(mt.get_proof(coin_file, new_root))
```

### Validation

check_validity returns true iff the supplied transaction, coin file, and merkle proof are each well-formed, valid, in-agreement, and destined for the local wallet.

```
check_validity(transaction, coin_file, mpop):
```

and

```
      valid_transaction? (transaction, coin_file)
      valid_coin_file? (coin_file)
      valid_mpop? (mpop, coin_file)
      acceptable_transaction? (transaction, coin_file)
```

`valid_transaction?` returns true iff the transaction was signed by its sender, and the supplied coin file corresponds with the transaction.

```
valid_transaction? (transaction, coin_file):
```

and

```
      signature(transaction.sender, transaction) == transaction.signature
      coin_first_block(coin_file) == transaction.coin_id
      coin_hash(coin_file) == transaction.coin_hash
```

`valid_coin_file?` return true iff the coin file has a valid origin (‘basis’), and contains only valid transaction blocks.

```
valid_coin_file? (coin_file):
```

and

```
      valid_coin_basis? (coin_file)
```

```
valid_transaction_blocks? (coin_file)
```

`valid_coin_basis?` returns true iff the coin file was a member of the genesis block, was created as a reward for transaction processing, or contains a valid third-party payload.

```
valid_coin_basis? (coin_file):
```

or

```
map [launch? grow? external?] coin_file.kernel
```

`valid_transaction_blocks?` returns true iff every block in the coin file is valid.

```
valid_transaction_blocks? (coin_file):
```

and `(map valid_block? coin_file.tx_blocks)`

`acceptable_transaction?` return true iff the the coin belongs to the sender and the transaction is destined for the local wallet.

```
acceptable_transaction? (transaction, coin_file):
```

and

```
transaction.recipient_id == me.wallet_id
coin_last_block(coin_file).transaction.recipient == transaction.sender
```

### Merkle Proofs

`valid_mpop?` return true iff the supplied merkle proof for the coin file is valid.

```
valid_mpop? (mpop, coin_file):
```

and

```
valid_roots? (mpop)
valid_proofs? (mpop)
fully_saturated? (mpop)
leaves_match? (mpop, coin_file)
```

`leaves_match?` return true iff the leaves for each block in the coin file match.

```
leaves_match? (mpop, coin_file):
```

and

```
each_hash_matches? (mpop, coin_file)
txs_align? (mpop, coin_file)
```

`each_hash_matches?` return true iff the leaves in the merkle proof match the transactions in the coin file.

```
each_hash_matches? (mpop, coin_file):
```

and `(map eq? (zip mpop.leaves`

```
coin_file.tx_blocks.coin_hash))
```

`txs_align?` return true iff each transaction in the coin file has a matching pair of entries in the merkle proof, and the coin is not missing any transactions present in the proof.

```
txs_align? (mpop, coin_file):
```

and

```
    map (lambda(tx_block, leaf: eq? (H(tx_block) leaf)) (zip mpop coin_file.txs))
    eq? length(coin_file) length(mpop)
```

### Validator Selection

`coin_validators` returns a deterministic list of nodes responsible for the validation of the supplied coin.

```
coin_validators (TREQ):
    mapcat H(CMR, TREQ.coin_id) H(CMR, H(TREQ))
```

`valid_povv?` returns true iff the supplied proof of valid validator can be verified.

```
valid_povv? (povv):
```

and

```
    povv.before < povv.address < povv.after
    eq? povv.root H(povv.before, povv.after)
```

### Conclusion

We have described a value management athe network packet level utilizing p2p self-validating atomic units of value to improve the security and efficient scalability of any system built on it while preserving confidentiality by design and p2p interoperability without having to depend on third parties. The current suggested block time of 1 minute can handle up to 800 Billion devices and throughput of 60 Billion transactions per minute (per block). In exchange implementations will give up the guarantee that they can query a ledger for an account's balance. This is an important ability for some use cases, and in those cases Implementer can partner with a ledger or even move the ledger onto Toda. So yes you can use a ledger in Toda removes the dependency on the ledger.

The resulting system is as decentralized as it can get using PoW where Work is Distributed Computing limited by the speed of light to ensure devices are dispersed across the network, every active user is a node, instead of a replicated ledger. In a replicated ledger all nodes have total knowledge of the entire system, but in TODA each individual node only knows about its own proofs and those of its direct neighbours in V1 implementation. However in V2 they will have some neighbors and some as elected by the pseudorandom function based on the last 2 merkle roots, but given it is stateless and information shared are based on Hashes and not necessarily the information itself, the system remains confidential by design. It is extremely important to note, that through every one block of time, there must be one single hash replicated through every device and

continues to chain indefinitely.