Fight the dust in the IOTA Tangle

Adrian Ihly (iota@ihly.net)

June 26, 2020 – Version 1.0

Abstract

The IOTA DLT tangle (see http://iota.org) has as many other DLT the problem that a large number of addresses can be created, aka as "dust", and especially in IOTA with a very little costs, a minimum of 1i and some little prove of work (PoW) investment. Adaptive PoW, Mana and sharding may delay, but not eliminate the problem. If fractions if i's (i.e. for colored coins) will be allowed one day, the problem even accentuates. In short, dust has the potential to destroy the IOTA ecosystem. However, low value transactions and balances are key for IOTA as a micro transaction platform.

Creation of a non-zero balance addresses must come with some cost beyond PoW. The idea is to require a minimum balance for each address, either by directly reaching the minimum balance or by borrow the remaining IOTAs from another address. This way, any low balance and low value transaction are still possible, while there is always a deposit locked until the address reaches the minimum or a zero balance.

All costs are fully recoverable by the lending owner, nothing flows to miners or into other channels.

The minimum balance can by dynamically adapted to the network needs to allow all use cases while keeping the number of active address and snapshots at a reasonable size. This is valid for a single root tangle or future sharded breakdown. It will also allow future handle fractions of IOTAs i.e. for colored coins.

The system would only affect a very small number of transactions, most transaction will not be affected at all. The complexity can he hidden in node and wallet libraries and only very few additional parameters and functions are necessary.

In short, the dust problem can be addressed at the root.

Problem

The IOTA tangle allows to create practically infinite number of addresses with very little amount of value on each of these addresses in microtransactions for a very little effort of PoW, in theory 2'779'530'283'000'000i can be distributed to as many addresses. If fractions of i's will be allowed one day, i.e. to enable especially colored coin use cases will open even more the opportunity for attackers and bad use cases to open too many non-zero addresses that needs to be tracked by nodes and snapshots.

2.7*10^15 i's * 81 trytes for each address and some bytes for roughly at least 50 bytes of storage, an attacker may enforce a snapshot to use up to 135 petabytes of space. For one petabyte of address space, an attacker would only need to source his IOTA base and invest $4.2Mio with today's market value, and possibly a somewhat higher value to inject the distribution transactions. While this seems to be a lot of money, this can be worth for an attacker to extort an ecosystem or promote an alternative chain. The

IOTA ecosystem will break apart way below the storage need of one petabyte, a practical limit will currently be more like around one terabyte or below, so attacker can reach his goals for a fraction of the costs above.

The cost to create i.e. million or billions of non-zero balance addresses must be significantly higher than the storage costs multiplied by the number of nodes on the tangle or withing a shard.

On the other hand, low volume transaction and low balance addresses are key to the IOTA system success. It allows many use cases that are not possible with other DLT or traditional value channels.

Colored coins may need today to pre-allocate many IOTAs to allow all future business cases, like splitting the values. Once the dust problem is under control, IOTA may allow fractions of i's to allow splitting a colored coins and smart contracts may allow.

As IOTA is a permission- and feeless system it should not allow that costs flow to other channels like miners or transaction handlers. Costs should in a form of deposit that can be recovered once the address is not anymore used or stands for its costs itself.

Solution proposal

Creating of an IOTA address with a non-zero balance must come with some costs. However, as IOTA is and should keep a permission and feeless system, the costs should be fully recoverable.

A cost can be attached to an address when there is a minimum balance requested per address. For example, we can require that an address must have 1000i as a minimum balance. Any transaction that will leave an address not reaching the minimum balance would be rejected.

However, how can the tangle still allow low balance addresses and transactions? The key is lending. An address can needs to lend from another address and this other address can warrant for the missing IOTAs.

For example, a transaction move 1i to a new address ("A"). Another address ("W") lends the missing 999i to A to get to the required 1000i. W most have at least a balance of free (yet unlocked) 999i and these 999i will now get locked up on W.

W needs to sign with its private key the lending. W cannot spend these locked i's until further transactions happen on A. W can lean his balance, for example 1'000'000i to multiple other addresses until all i's get locked up. As W in any case needs to expose its signature multiple times in the tangle (once or more for the lending, and finally to transfer the released balance somewhere else), this will need reusable addresses as planned in Chrysalis. Together with the balance of A, a reference to W is stored with this address A on nodes and snapshots. Nodes only need to keep a such references where a lending is needed.

Over time, A may receive from anywhere more IOTAs and increases its balance. For example, an account ("X") wants to transfer 9i to account A. X does not need to worry about minimum volume, as the address exists with a non-zero balance, so there is a lending/warrant already present for the difference. After the transaction, A has now a balance of 10i, and W has now locked only 990i, so it got 9i released with this transaction as free IOTAs these maybe reused somewhere else.

Once A reaches a balance 1000i as the minimum volume, the locks on W for A are completely released and can freely use his balance. Nodes and snapshots can drop the reference on address A to W.

The similar happens when A gets to a zero balance, i.e. A transferred his whole balance to another address (“B”). When A gets a zero balance, the lock on W is released. A does not need to list anymore in further active addresses snapshots. However, if B itself does not reach the minimum balance after this new transaction, a new warrant (“WB”) need to be found for B, else the transaction will be rejected.

Optional: A possible new zero-value transaction to A type of new warrantor (“W2”) and signature of W2 will allow to transfer the current lending from to A from W to W2. This allows to transfer the lending from W to W2 in case W wants to unlock his volume for any reason. But possibly there is no real use case to transfer a warrantor, but a solution can be implemented.

A receiving address (“R”) owner is typically also the owner of the lending W address. This way, the owner has full control on the lending and can free up locked volumes anytime. But there is no requirement that lender and receiving address must have the same owner. There are valid use cases (i.e. colored coins below) where this is not the case. The tangle is agnostic to this.

Transactions that leaves at end any involved address with at least the minimum required balance do not need to worry on leading and does not need to provide a warrantor’s signature. Most real-world value transactions fall into this category. However, if an atomic transaction ends up with address below the minimum required balance and no valid lending, the whole transaction would be rejected by the tangle.

A new warrantor address may be part of an atomic transaction, for example with a minimum balance set as 1000i, address A may be start with 1i while warrant address W may also get an initial balance of 999i in the same atomic transaction. W can be a new address or an existing non-zero address already. A and W can be created as pairs, but do not need to.

The minimum required balance can dynamically adapt over time. For example, an address may always require a deposit of $0.001, the minimum required balance would be ~4’500i. A oracle process may find weekly a new market value and set a new minimum required balance.

Nodes may need to keep a little history of current and 1-2 past minimum requirements, as they need to validate transaction created short time back where another balance requirement has been set. In grace period (i.e. 5 minutes), a validator may take the lower requirements to validate. Once a transaction is confirmed, the transaction stays valid regardless of a potential future higher minimum. But any new transaction will again validate the address plus lending balance and needs to meet the current new balance, else the new transaction will be rejected.

So, it is maybe wise to allocate W with a slightly higher balance than currently needed. W may get a balance of 1100i in the above examples, even when only 999i are initially locked and 101i still free. When A receives later new 9i from somewhere ease and the new minimum is meanwhile 1050i, A can receive the 9i with the need of the foreign spender to worry about lending. After the transaction A will have 10i, W will have 1040i locked, and 60i still free.

The locked i's on W will dynamically change with each change on the minimum and with any value transaction made on warranted addresses. If the minimum requires for all lending of W suddenly more that the balance of W, the previous transactions do not get invalidated, but any new transaction with an involved address will require a new lending to get accepted.

Use cases

Use case A (“UC-A”): a business requires the transfer of a value from A to B that is way above the minimum, i.e. the transfer is a value of $1. The tangle works like before: A transfers about 4.5Mi to B and places the change value to C. C of course also needs to reach the minimum requirements, nut typically not a problem. In the rare case this would not reach the minimum, A need to provide more input to the transaction or a lending address signature, all handled in the background in the wallet software. Users do not recognize any of this process.

In the rare case a user with an account balance below the required minimum cannot move its values anymore without the help of another account. Another account may pre-fill a destination address with the minimum requirements, so A can transfer all free values from his account.

Use case B (“UC-B”): Service “S” offers a service that is payed with micro value transaction, where each transaction is likely below the required minimum address balance. S therefore publishes a prefilled address A with minimum balance (plus a little reserve, as minimum may be lifted soon) already on it. Consumer “C” of the service transfers a small value to A and does not need to worry on minimums on A. A consumer account needs to have at least the expected service costs + one minimum balance (plus a little reserve) as total balance to allow a change address to receive the change anytime.

After a while, S publishes a new prefilled address B, and can use the full amount on A for other use.

Use case C (“UC-B”): An company “C” creates in a genesis 1000 shares of a company as 1000i colored coins. It creates an address A with the 1000i and a warrant W to reach the required minimum.

Investor “I” buys one of these shares and advises to transfer one colored i to address B. C transfers one i from A to B, while it provides a warrant address W with a signature. W warrants for the remaining i up to the minimum required (i.e. 999i). C must in this case in advance carry (lock) the cost of minimal balance with a new lending on address B. This is typically not a problem, as the share represents typically a much higher value in the real world. As soon as “I” further trade his share (B is now zero balance), the warrant on B is released, and C recovers his advanced money.

In this scenario, IOTA tangle may even allow fractions of IOTA transactions and balances, in case IOTA will allow this in a future extension. For example “I” may sell half of his share on B to another Investor, lets say with investor I2 on address D and keeps the other half address C. “I” and I2 will have each a balance of 0.5i, and “I” need to warrant W1 and W2 for these 2 addresses D and C (W is released by then). The shares can now be split any further. This would eliminate the need to allocate many i’s to cover all possibly future trade scenarios (or recolor coins). Basically, an ICO may start with single 1i as a represent of the whole company.

The colored coin scenario is one case where the address and the warrant does not necessary belong the same owner. The warrant address maybe locked outside the control of the owner if the warrant address. In this scenario, the warrant address will have a balance with the exact difference to the minimum. So the owner of B is enforced to release the lock on W on any further transaction he does. In case the minimum decreases over time, W can spend the part of freed up i's to another address, so again, he can enforce B to release all lock on any new spending transaction, even when only fractions are spent.

Warrantor addresses may lean missing IOTAs to one or more addresses, this this depends on the business case and how the locks should be best managed.

Implementation

Nodes and snapshots need not only to keep track on address with their balance, but also a reference to a warrant address in their store, if case there is a need to.

Note: Only for a small fraction of addresses such a reference is indeed needed. Most addresses anyway reaching the minimums and no reference is needed. For an address just reached the minimum (with a transaction or lowering required minimums), the reference can be dropped.

Nodes also need to maintain an index where a given warrant is referenced. A function should provide all addresses where a given warrantor is registered. Addresses that comply with minimum balance or have a zero balance does not need to be returned.

For every such warrantee address, the difference of the actual balance to the minimum required (or zero in case the actual balance is >= minimum required, or the address has a zero balance) needs to sum-up to a total locked balance. If the warrantor address balance is higher than this sum, the difference can be used for other warrants, or freely spend as input in transactions. If equal or below (i.e. due to lifted minimum requirements), the full balance is locked up.

Transfer transactions is a send_transfer API may need an optional parameter for each proposed transaction structure of a warrant address. The send_transfer API need be able to sign a warrant, therefore the private key must be accessible to this address.

Sample:

```
    tx1 = ProposedTransaction(
          address=destinationAddress,
          message=None,
          tag=None,
          value=1,
          warrant=warrantAddress)
```

The warrant parameter is optional and maybe None if no lending is needed. For zero value transaction (messages), no lending is needed in any case.

A warrant address may sign the transaction hash. The signature should not discover part of the private key, so ECC or other signature technique should be used.

Send to the tangle may include a new parameter for a warrant for the change address:

```
    send_transfer(
          transfers=[tx1],
          inputs=[foundingAddress],
          change_address=changeAddress,
          change_warrant=changeWarrantAddress)
```

The change_warrant parameter is optional and maybe None if no lending is needed.

A node validator then checks on after the transaction:

- All addresses reach the minimum required balance, either by a balance on the address, or by a combination of address balance and warrant. If not, the whole atomic transaction is rejected.
- Any warrant in the transaction must have at least a balance of all refenced addresses. Therefor a node needs a function to quickly find all balance differences of a given warrant address.

Some helper functions may be needed for wallets:

```
        current_minimum_balance_requirement()
```
The API returns he current valid minimum balance requirement in IOTAs. In phase 1 & 2 (see below) in the transition, this requirement is returned, even when not yet enforced.

```
        locked_value(addresses=[address])
```
The API returns an array of locked balance per address provided.

```
        active_lendings(addresses=[address])
```
The API returns all details of all lending for all addresses provided. In the details per provide assets is an array of warrantee addresses with their difference from their balance to the current minimum requirement. Warrantee addresses where no lending is needed anymore will not be listed anymore.

Note that validator need to use the minimum required balance of the time the transaction has been injected. In the meanwhile (a few seconds or minutes), the system may have a new minimum requirement, and this should not invalidate transaction that would have been valid at the time injected. So, they need to keep a little history with timestamps of these requirements.

Snapshots need to keep any non-zero balance address, current balance and a reference to the warrant address, if the address does not reached the current required minimum and a warrant address is established. Else the warrant address can be omitted from the snapshot. Only a small fraction of addresses is expected to have a need for such a refence.

Mana may include the warranted balance into the reputation calculation and uplift low value transactions in mana value. Details of this idea are not discussed further in this white paper.

Sharding may keep warrantor addresses in the same shard as the warrantee. However, some future challenge may need to be solved when sharding will be introduced. Each shard may implement an own oracle that defines own required minimum balances depending on the shard's need.

The oracle to define a minimum requirement should dynamically adapt to the network needs. It may takes references from the real world like current Mega-IOTA price, current snapshot database size, cost factor of storage and other parameters. Changes should be smooth (and therefore regular, i.e. weekly) and to some level predictable to not unexpected break typical use cases. Warrantor addresses should keep some free IOTAs to cover increased minimums, except when they want to enforce an release on a next spending transaction on the warrantee.

Introduction and transition

Preconditions are reusable addresses where a verifiable signature, for example Elliptic Curve Cryptography (ECC). This extension is planned in the Chrysalis release IOTA 1.5.

A lending system can be introduced in 3 phases.

Phase 1:

An oracle service will provide a fixed or a dynamic minimum requirement per address.

Phase 2:

Client nodes may start work with the protocol extension that allows to include warrants into transactions, even when this currently does not have yet any consequences when not providing minimum balances or warrantors.

Validators node may check the new requirement but still accepts transactions when they don't provide leaning or underfunded leans. The tangle stores a warrant address into the tangle if provided, regardless if they provide enough missing IOTAs or not.

Over time, many low volume addresses have now a warrant address reference, even when the warrant may or may not have the required balance. Service provides can start given all warrants needed when the system activates and provide uninterrupted service for phase 3.

Reports may be built to review the number of addresses that do not yet comply to the new rules. On a given threshold or event, phase 3 may be activated.

Phase 3:

The system goes live, and validators reject transaction the end up with too low balance.

All existing transaction and balances are still valid, but any new transactions need to meet the new requirements.

If an attacker may have already created billions of accounts, these may be outsourced into a single, trusted database. The tangle does not necessarily know what real world transactions are and what are attacker transaction. If anyone wants to recover his coins, he needs to transfer the low volume address to a new address that complies with the new rules. Non transferred addresses may be dropped one day, if costs of storage is higher than potential value behind these addresses.

Conclusions:

- Dust is a potential lethal problem and may not be addressed with existing throttling logic
- Creating dust comes with significant (temporary) cost to make unattractive for an attack or strange business cases that uses too many active addresses
- Normal value usage of the addresses above the minimum will free of any hassle like today, users won't need to worry on minimum requirement on addresses. If when the rare cases is needed, wallets can handle this in the background.
- The temporary costs are prepaid and is fully recoverable. Typically, the owner of warrant address also controls the warrantee and can anytime release his locked-up IOTAs
- Service provides may provide pre-filled addresses for micro transaction; those do not need to worry on reaching the minimum requirement
- Nothing is lost or flows into to foreign channels, like miners. This is large deviation to other DLT systems where costs are typically on the sender side and are not "lost" to miners
- Snapshots will not get significantly larger. They only need to keep a warrant address for addresses that do not meet minimum requirements at the time of the snapshot (or nothing if there no historical warrant in the tangle)

- A dynamic adaptation (details not part of this while paper) keeps the system resources to run a node always into acceptable limits
- Colored coins use cases are still possibly. Even fractions of IOAT may be allowed with future protocol changes, without the risk of new dust attacks
- A seamless transition to be possible with only little changes in the protocol. Wallets and nodes need to update their software in phases
- Code changes needed to enable this feature will be moderate. The processing overhead is minimal for a node and validator. Most transactions are anyway not affected.
- Some research and simulations are possibly needed to the required minimum oracle.
- Sharing, Mana adaptions etc. may be another field of research, however there are indications for straightforward solutions

In short, cleaning the dust is just a few code lines away.

Adrian Ihly
3065 Bolligen
Switzerland
iota@ihly.net