Dito
-
Do your Inputs + Outputs

Coju Lapa
cojulapa@tutanota.com

February 2, 2022

Abstract

This paper is work in progress and just another crazy idea for mutual credit system.

1 Introduction

Like Ripple[1] and LedgerLoops[2] the proposed system is based on trust networks. Trust networks are built by nodes which have trust relationships. This means they accept a temporary in-balance of credit.

A characteristic of such systems is that every node taking part can selectively decide who can back existing credits. In fact each node can set the limits to which other nodes will be able to have a negative balance. There is no obligation to trust all, or better, all in debt, that they will be able to provide value for your credits. In today's monetary systems most participants are not asked about the creditworthiness of debtors. Instead, these decisions are made by a minority while consequences show up collectively.

It is clear that one cannot establish trust relationships with the necessary amount of people to keep up a complex economy like ours. Therefore, the challenge in such a system is to enable fast payment options between nodes who do not trust each other directly but are connected with a trust line.

In LedgerLoops the route finding of a trust line is done in the moment of payment. This is a moment when wait times are hardly accepted by users. Moreover, at this point of time all nodes in a loop need to be active to forward the request.

Ripple solved the trust line problem with one entity having a birds eye view of the network. However, this requires one entity to know all trust relations, a clearly intimate information and this also makes it somewhat centralized.

Dito, wants to use the time between issuance of a coin until it's spending for the necessary line allocation. Considering that this time is generally anyway quite long, at least long in terms of IT, this seems to be a potential that should be used. This strategy will also help to handle irregularly active nodes joining the network without a central entity organizing the system.

A general problem when bringing the concept of money into the digital domain is the fact that all digital data can be duplicated easily. This results in the double spending problem. Each

[1] https://ripple.ryanfugger.com/
[2] https://ledgerloops.com/

digital payment system has to face this issue. Dito tries to use the strengths of information technology rather than trying to circumvent the problems they cause, when digitizing the typical concept of notes and coins. Those strengths are: simple duplication of data and therefore practically everything that exists in the digital domain, considerably large memory size and fast data comparison. The proposed mutual payment protocol exploits the facts that coins can be created with little effort and their information spread easily.

2 General Considerations

Nodes which trust each other are direct neighbors in a network like in Figure 1, indicated by connecting lines between them. All connected nodes form a trust network. Trusted nodes keep their own ledger where they keep track of the current status of "I owe you" (IOU) from each other. This IOUs are meaningful simply because of the trustful relationship they want to keep up. To make use of this potentially accepted range of IOUs in the following, the two nodes will hand out conditional IOUs. These IOUS will only be valid under certain conditions and can be understood as "I will owe you if..." These conditions must not be publicly visible to ensure privacy. Accordingly, two trusted nodes must establish an encrypted communication channel with each other. Over this channel they can hand out and accept conditional IOUs.

Nodes which do not have any trust connection to other nodes inside the network can not trade with this respective network. This makes sense because if nobody sees anybody in the other group as trustworthy they should not accept a temporary in-balance only backed by the believe they will get something back.

If nodes, like Ann and Toni in Figure 1, want to trade, there exist a trust line between them. As a result, they will be able to trade although they do not trust each other directly. This means they can trade without accepting further risk or more precisely without going out of their comfort zone of whom they decided to trust. Activating this trust line is the tricky part in federate bookkeeping.

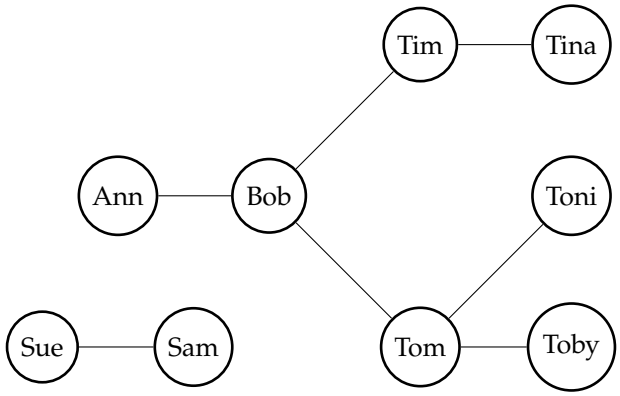

Figure 1: *Ann, Bob, Tim, Tina, Tom, Toni and Toby form a trust network. However, Toby would for example not trust Ann. Sam only trusts Sue, so those two could not trade with the other network yet.*

It shoud be clearly mentioned that whether you will be screwed or not primarily depends on your risk management with your direct neighbours. Expressed differently, your susceptibility to losses is determined by the accountability of your trusted nodes and the limits to which you allow them to spend first. Since it is always clear who is backing your credits. Insight into the nature of humans is therefore of utmost importance to make this system work for you. In contrast to other

common currencies, where value to the currency is backed by the assumption that there will be somebody willing to work for it in the future, here everybody knows exactly who is backing their credits.

3 Simple Example Story

In this scenerio Ann has a huge amount of apples. Bob does not have any, so Bob asks Ann to get some apples. Bob is in good realtionship with Ann, so he wants to give something back. Unfortunately, Bob does not really have anything to offer to Ann. Even worse, he does not know if he, himself, will ever be able to provide something of interest to Ann.

But maybe some friends of Bob might have something valuable to Ann. So Bob still tells Ann that he would really like to help out in case there is an opportunity.

Ann now creates riddles to which only she has the answer to and sends this to Bob: "If Bob comes up with the solution to one of this riddles: RA1, RA2,... than Bob has returned the favor."

Bob knowing that he probably won't have anything of interest to give to Ann, asks his friends for help. He sends the following messages. @Tim: "If Tim can tell Bob the answer to riddle RA1 or RA2, then Bob owes Tim a favour." @Tom: "If Tom can tell Bob the answer to riddle RA3, RA4 or RA5, then Bob owes Tom a favour." Tim and Tom send the following messages to Tina, Toni and Toby. @Tina: "If Tina can tell Tim the answer to riddle RA1 or RA2 then Bob owes Tim a favour." @Toni: "If Toni can tell Tom the answer to riddle RA3, then Tom owes Toni a favour." @Toby: "If Toby can tell Tom the answer to riddle RA4, then Tom owes Toby a favour."

We should mention that the riddles are practically unsolvable for Bob, Tim, Tom, Toby as well as Toni, and it therefore lays in the power of Ann to hand out the solution to enable Bob to let go his debt. This is important because Bob wants to know when he has returned the favour to Ann! Since the riddles are unsolvable Bob knows that, when the riddle is solved, Ann has received something in return for her apples, because she has revealed the solution in a trade.

But more importantly, Bob wants to know WHO has returned his favor to find out whether he can do something good to this person. So that finally he can really do his own duty in returning the favor. For this let's check out how Ann uses her coin backed by Bob's favor.

One day Ann goes to a bar and wants to drink some Tonic. Before paying they agree to check for open favors. They find out about the open favor of Bob. So Toni now says "If you tell me the solution of riddle RA3 then you can have the drink on Bob." Ann reveals the secret and can enojy the drink. Toni then informs Tom about the solution, so Tom now owes him a favour. It is Toni's choice whether he sets up riddles just as Ann did with Bob. This could be a good option if Tom does not have anything of interest to him or if Tom is lacking behind in the private ledger between them. Tom, now also aware of the solution, can go to Bob and likewise ask for his favour or give out riddles.

4 Technical Details

Riddles

The riddles need to be of a type (like SHA-256) where the effort in finding the solution stands in no relation to the effort of redoing the favor but everybody can easily check if an answer is correct. Nobody will really try to solve the riddle on his/her own. So Bob is only even if Ann reveals the solution in exchange for something in a trade.

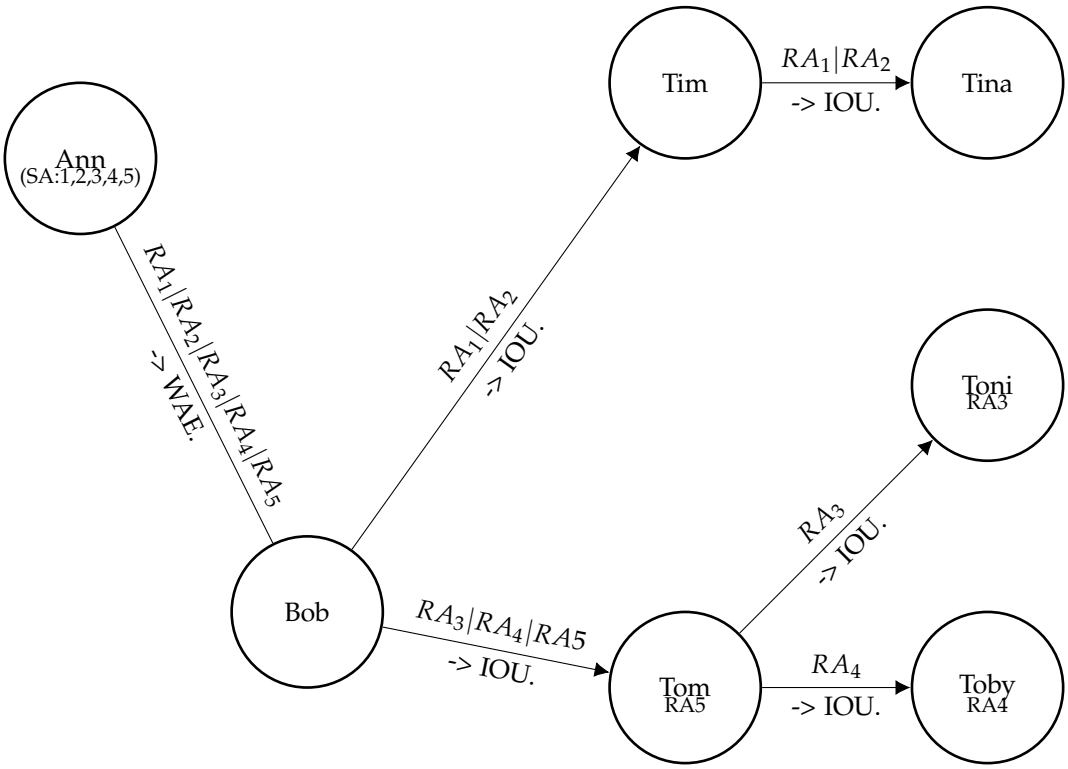

Figure 2: *The conditional IOUs in the net. All statements are meant to be spoken by the sender, which is the origin of each arrow.*

To prevent Bob denying his dept at a later stage or if he simply forgets (data loss) about it, Ann could ask Bob to sign his promise. Like this, she has something in her hands to show him and make him aware that they are not even yet.

Creating riddles and giving them out is the analogy to minting your own coins. In the above example Ann minted her own coins but Bob gave those coins value, by asking for acceptance (distributing the riddle among his contacts) and backing them (I will owe you if you do something for Ann).

Needless to say, that coins can not only be minted at the instance of a physical trade. Two parties can exchange and spread coins prior to a physical trade to create liquidity along trust lines. The difference would be that the issuance condition would not be "we are even if I can spend that coin" but "I owe you if I can spend that coin".

Acceptance

For the process of payment, each coin has an identifier. If the network is once flooded with issued coins the probability becomes high that one node will accept the coins of another node. The process of checking for acceptance compares the list of coin IDs on the payers wallet side (where the solution is known) with the coin IDs the other node was offered conditional IOUs and which were put in the acceptance list.[3]

The reveal of the solution requires a lot of consideration since this is the only point of time

[3]For higher privacy one might only share a fraction of digits first, and then consecutively narrow the search.

with direct interaction of two non trusted nodes. Even if Toni publishes the riddle's solution right after receiving only people who are in the trust line can claim their conditional favors. They have earned it.

For the unique identification of each line, a node who plans to accept a coin needs to keep one riddle for itself. If a node does not plan to accept, it can function as a relay station, forwarding all riddles without reserving one for itself. Forwarding all riddles makes sense when a node does not plan to offer any services. Or in case the node has already built up a considerable amount of coins and trusted nodes are lacking behind redoing their favors. Then it makes sense just to forward to allow for resolving.

But what if Ann tries to use the favor a second time? Going to Tina and reveal another secret? Before Bob has taken notice of the solution coming from Tom? So that he wasn't able to pull back his conditional IOUs with Tim and Tina?

Allow Double Spending

Note that, Toni or Tina would not be harmed by double spending, as long as Bob will step in for Ann, and do a favor in advance for her. So that he already earned the next portion of apples. So indeed, Ann can spend the same coin again using another riddle. However, we want to note that these claims will fall back on her friend Bob. Bob might want something in his hands to keep Ann responsible. So Bob could have asked Ann to include a phrase like "If more than one solution would become public I, Ann, will owe Bob, the additional amount of known solutions". If this is signed by Ann, Bob can make Ann accountable for overspending. Like this Bob can make sure that Ann does not benefit from double spending. At least not without paying back at a later stage. If she uses them, Bob can claim this unexpected flavors from her. Tom of course would do the same if he forwards more than one riddle. Setting up the same contract with Bob, he can ensure that all in all he does not have to face a deficit. To a certain extent this option might be helpful to bring liquidity. Again this is based on the fact that there is trust between direct neighbors. The disadvantage of this liquidity option would be larger possible risk spans. Since the risk span of Bob if he forwards lets say 30 riddles from Ann and distributes this coin with his CIOUs, is 30 times the value of one coin. The only collateral that Bob will get something back is trust in the capabilities and skills of Ann. It should be emphasized that Bob has control over the amount of total claims that could potentially come back on him. He can decide how many riddles to give out to which channels at which time. He can use this to manage financial risk among his trust nodes.

Block Double Spending

To overcome the possible high risk spans when multiple RA riddles are created and distributed, another riddle, RB, can be included in the coin. Revealing the solution to this riddle marks that the coin should not be accepted any more.

For this to work we need to add a public key of the owner in the coin information. When the owner wants to spend and has chosen to block double spending with the issuer of the coin, the RB riddle solution needs to be published together with the coin ID and the RA signed by the owner. What is made public knowledge is which coin (ID) was spend (RB) at which trust line (RAx). Having this information signed by the owner makes sure that no other node could take the now known RB riddle and take another RA to publish it again. Everybody in the network can help to update and distribute used coins and their riddle solutions.

Generally, Toni only needs to get the signed block from Ann which fits coin ID, his RA solution and the RB solution. It is true that Ann could still publish another signed block. To relatively certain that Ann can not double spend Toni needs to wait a little until he listens back that the

majority of the network has adopted the spend information of the coin in his trust line. For this purpose it might be helpful to have another communication network which is used for this information spread. In this communication network also non-trusted nodes can exchange information. Since this information can only prevent everybody from fraud, or at least it is always more precautions to keep track of all information about what one should not accept. It is worth to mention that the entries in this list do not depend on their time order. The coins in the list have been spent. It doesn't matter when and which coin first. They should not be accepted any more. Of course nodes are meant to keep the first signed resolution of a coin they see. (Comparable as in a blockchain everybody is meant to keep the longest chain.) Here everybody who has the coin on their acceptance list has a subtle interest in spreading the correct news so that nobody in their trust line accepts the coin again. Which will cause troubles since Bob was not willing to back multiples of the coin.

4.1 Optional

Callback

For practical reasons we might want to introduce an RC riddle for cancel or callback. Publishing this would render all contracts which include the involved coin ID invalid. This might come in handy in case of an emergency. But could also be used to cancel spreaded coins if the owner wants to reset them to maintain privacy, in case the coin ID has already been shown to many strangers. Or when the owners system has been hacked. If the attacked account quickly realizes it by revealing the SCs the stolen coins can be made worthless. To renew the coins after the attack, the contract with the direct neighbors which issued the original coin would contain "If SC is revealed, the current coin becomes obsolete and a new riddle task, will be sent as replacement."

Opt Out

If the double-spending option is chosen it makes sense that also Bob or all in the chain have a trigger to pull back, lets call this the RD key. Then Bob or anybody in the chain is able to stop the spending of Ann when their private credit ranges with their neighbor nodes come close to their limits and they start to feel uncomfortable. However, lets emphasize that this is primarily a signaling function. And nodes having this coin in the list should take it as a good recommendation to deny accepting it.

A Note on Misbehavior

In communities criminal activities, like double-spending, can be prohibited by making them technically impossible. Or by detecting these actions and reacting in a way that gives the causer negative consequences. Which of these two strategies is more feasible depends on the effort needed to reach either bulletproof security (nobody can do it) or action tracking (who did it?). Since all responsibilities are known in federated bookkeeping, keeping actors accountable for misbehavior can intrinsically be achieved rather easily.

5 Data

This section should structure the data every node needs to keep.

CID stands for coin information: ID, Date, Value, optional message, value relation. SA, SB, to Sx are the solutions to riddle RA, RB, to Rx . CIR is the coin plus its corresponding riddles.

	myCID		oCID			
directory	Wallet	Safe	tFWD	AccL	FWDd	Blist
holds	CID, signed by I_{nodes}	ID + SAx, SB, SC	CIR + I_{nodes} , singed by I_{nodes}	Flag	O_{nodes}	SB/SC
use to	pay	verify	manage	earn	liquidity	overspending
accessed	spend diff	spending	forwarding	accept diff	calc potential risk	update & before accept!

Table 1: *Coin storage content*

The message of a coin would look as follows. The signatures are only relevant for the direct neighbors to keep. With them they can keep their trade partner accountable to the conditional IOU.

```
CIR = {
 "Id": "0x0008",
 "Value": "3",
 "Issued": "05.01.2021",
 "Pub Key: Pub Key of Ann" (in case double-spend should be blocked)
 "RC": "17e4615a2e2a318ea4220b32176eb3980e8ddcd3e47d10f56ce7a7b30a7c05a5",
 "RB": "17C486709819DB665D9439B72F97646C3C47A95E587578B8B1939AC053515DAF",
 "Value Relation": a bag of apples
   ------ part below is updated between nodes ----------
 "RA": "{177A858D134B0255BCF370408B14EE390E28A332811144C8E274BEA01667D7A3,
        5E35A10232F6BF6F4E94FD2AFDC3D3CB99DA5B30097728188ACCF7204A928A6C,
        EC39C3BD75E6BB9D9B35F5CF8071443E8254456BF6DF479707EF1BADF68AFD0C,
        3240E715B45C650BD113BEFBE04CE6E5D72F8B3DF826620B338BEE656EA5EAE5,
        C35E5BE7B5F11D8AECA3167CA5F69128DDD8695842A018443224A01C1F4A5D35,
        C894E31B7F17271001DC1F4D240D29DAD52A98147D0A622EA471556B9A46D042,
        }",
 "Condition":  If Bob can solve one of the riddles, we are even.
  If however more than one solution will be known,
  Ann owes Bob x, where x = (the amount of known solutions) minus 1.
 "Signature": "Ann"
 ("Signature": "Bob")
}
```

5.1 myCIDs

The myCID section contains the coins created by the node itself, this contributes to the positive balance of the node. For its purpose the node must provide secure storage, which is probably the most critical point. This is like your depot. It holds everything you have and can spend. For security purposes let's split it in two sections.

5.1.1 Wallet - w

The wallet holds the coin IDs and the signed contracts with the neighboring nodes. This data is used each time a node want to pay because the IDs need to be diffed with the acceptance list of the node who should be payed. It might be unnecessary for the coin owner to keep the riddles in the wallet, if asked for a solution, the node could check through the stored solutions, maybe this is a reasonable trade-off compared to the storage needed.

5.1.2 Safe - s

The safe holds the solutions to the riddles of a coin. It is only accessed when a matching coin ID, which was found during a payment-diff, should be spend. Showing the solution to the riddle finally verifies that you are the owner of that coin.

5.2 oCID

The oCID section holds all coin information that passed the node. This section is very important for risk management so that all potential credit exposure stays in accepted boundaries. Further, this list is used to only write relevant data into the blacklist.

5.2.1 tFWD, AccL, FWDd

The tFWD list holds the conditional IOUs, which will give a reward when a solution can be provided. The node can decide to accept or *to forward* riddles. When a riddle is reserved for own acceptance a flag is set in AccL. This is used to "earn money". Whereas when it is *forwarded* the net balance of the node will stay the same since it functions as relay station, only providing liquidity. For forwarded riddles it needs to track to which node which riddle was given. This is saved in the FWDd list.

So with each riddle either the AccL flag or an output node can be set. The AccL flag can also be canceled again to use the riddle again for FWDd. The FWDd can not be canceled that easily, only after a request down the network and resolving the contracts as long as they have not been activated.

5.2.2 Blacklist - b

The blacklist is the collection of the canceled/used coins a node became aware of. This list is frequently diffed with other network contacts to keep up to date. Most importantly is an update before accepting coins. Since coins have a maximum lifetime after which they need not to be accepted, older coins must not be kept any more and can be deleted from time to time to minimize data space. Since this data only protects the nodes from fraud this information is time independent. The order of entries is not relevant in preventing them to accept this coin.

Blacklist entries can be of the following type:

```
General form  = {"Id", "Date Issued", RX, SX, (RY), (Sy), (signature)  }

Cancle        = {"Id", "Date Issued", RC, SC,      , }
Resolve       = {"Id", "Date Issued", RB, SB,  RAx, SAx , signed spending node}
Trace back    = {"Id", "Date Issued", RB, SB,      ,  }
```

The problem of spam could maybe be omitted when every node only saves blacklist entries of nodes it has been offered. The issue date is still saved so that the list can be cleaned from old entries after a certain lifetime of coins.

6 Processes

The following flow diagrams should illustrate how two nodes have to communicate to realize the functionality. The whole operation can be broken down to these basic processes.

- Creation of a Coin / Pay with Futures

  The creation of Riddles and conditional IOUs, only possible between trusted nodes or at least nodes who give the other side the benefit of the doubt for a small amount to start with.

- Forwarding Riddles

  The node who wants to forward riddles first sends a request with the coin information whether the other nodes is interested in the riddle job. If so it answers with the amount of riddles it could make use of.

- Inbox Processing

  The inbox catches all incoming FWDd coins from other nodes. From time to time these coins need to be processed to manage risk, provide liquidity or generate a positive net balance.

- Spend a Coin

  The way to process a payment between non trusted nodes.

- Update Blacklist

  Updating the blacklist and with it the AccL about which coins should still be accepted.

When no match is found during a spend coin process, they either need to find an alternative payment option or give the other the benefit of the doubt and accept a CIOU from a non trusted node.

7 Functionality of an App

This section elaborates a little bit how an app should look like to make such a system usable. The actions a user would want to perform.

7.1 Depot - dipo

A function which visualizes the current status. Using the information from the wallet, the AccL and the FWDd list ranges possible and existing credit exposure can be calculated.

7.2 Add a New Trusted Node

Establish encrypted communication channel, key exchange via screen scanning.

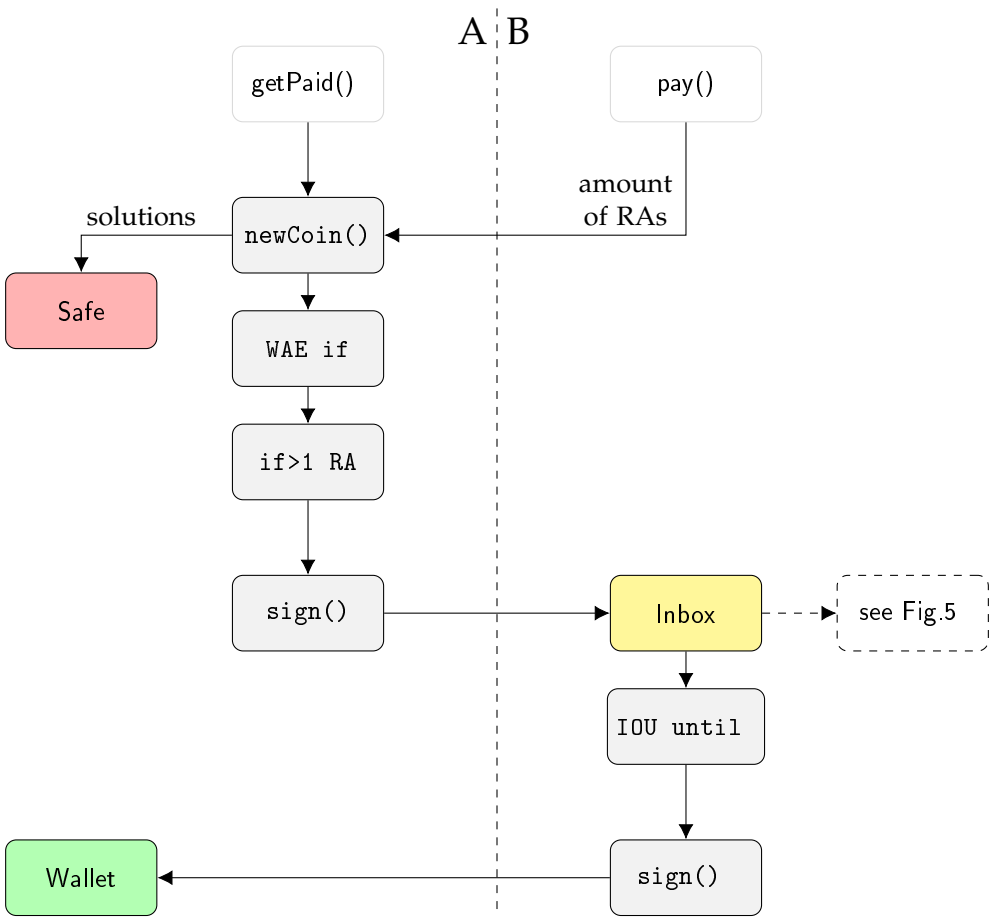

Figure 3: *Coin creation between two trusted nodes.*

7.3 Manage Nodes - dito

Allows to set the credit limits for all trust nodes. dito here stands for "do your input and outputs" Maybe also provide adjust settings how many riddles should be given out, whether incoming CIRs from a neighboring node should always be forwarded.

This section would leave a lot for personal settings about how the node behaves when processing incoming requests. Nevertheless, it might also be desirable to have a manual mode where every incoming request is viewed and managed by the user.

7.4 Trade Exchange

It would be handy if both partied could initiate a pay process.

7.4.1 Pay - o

Initiate to pay somebody and select from 2 Options:

- Use your past earnings(check for coinciding CIDs), relieve others from their debts
- Issue a coin, pay with future. Only possible if recipients trusts you.

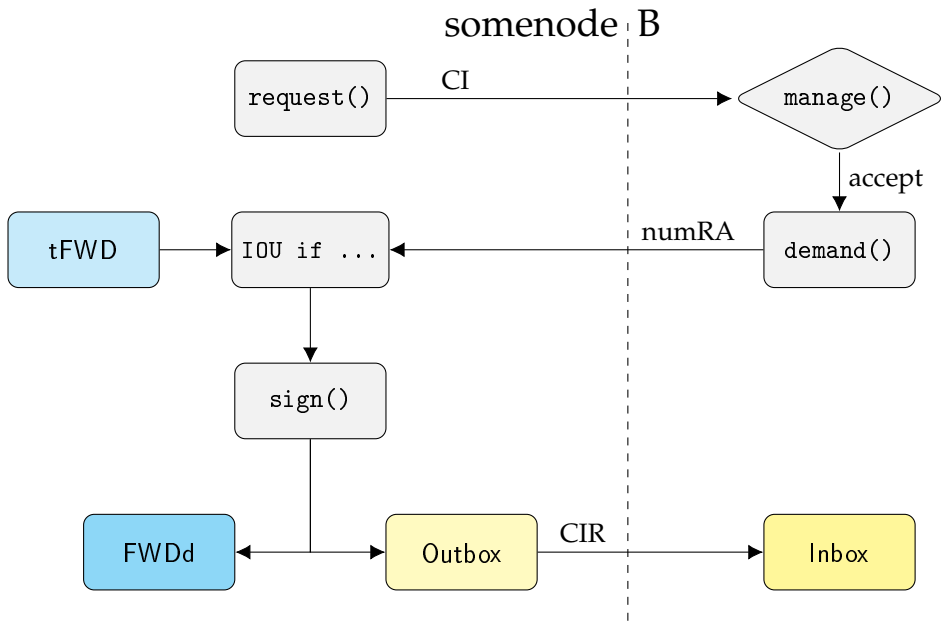

Figure 4: *Forwarding request.*

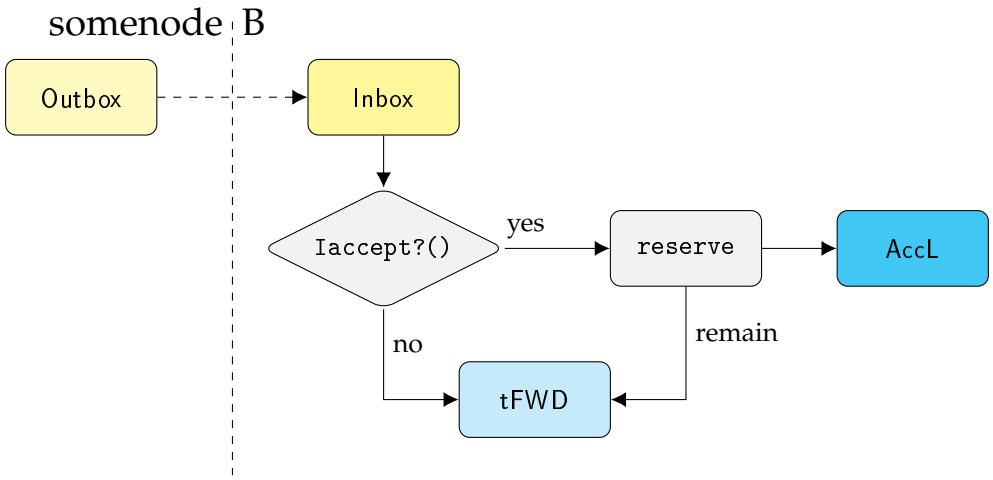

Figure 5: *Process Inbox.*

7.4.2 Pay Request - i

Initiate getting payed.

- By accepting CIOs (futures) of others (maybe as only this time, without adding this contact to trusted nodes)
- By letting somebody pay back the debts (check for coinciding CIDs).

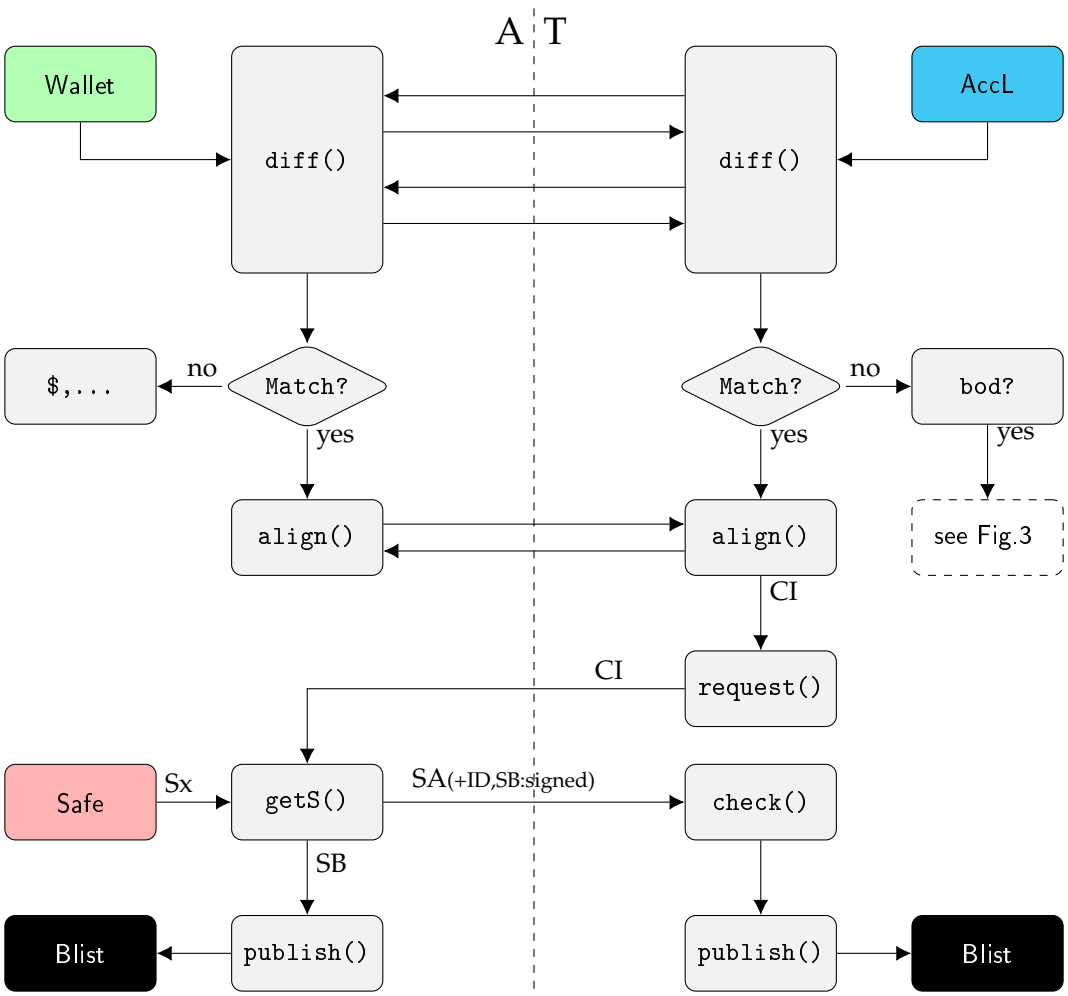

Figure 6: *Spend coin.*

7.5 Dead Nodes

Of course, it could happen that nodes do not become active any more. Own issued coins will simply not be used and other nodes accepting them will delete them when the lifetime expires. The only one lacking payment is the one in direct connection to the dead node. Assuming that direct neighbors are only made with trusted parties, the issuer should be able to contact the node(person) via a different channel (physical, phone, mail, ...) and request the debt. If no pay back can be achieved this lies in the responsibility of the creditor.

8 Some Thoughts

The cancel function of a coin could also be used to restart a coin to protect privacy if a lot of diffs have been performed already. The trade-off a node makes here is temporary liquidity against privacy.

The amount of riddles handed out, is a trade-off between general acceptance and the risk of

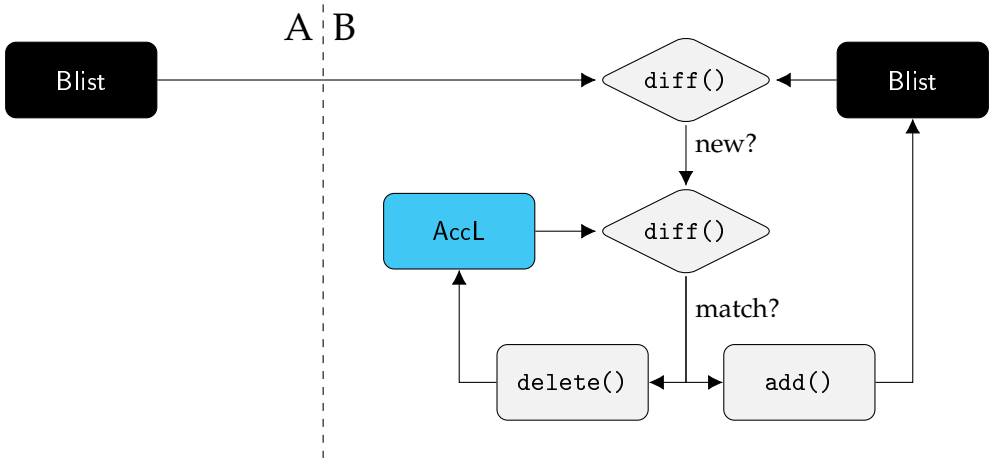

Figure 7: *Update Blacklist.*

owing your trustee the coin value times the number of riddles. This consideration is especially important if Ann does not use multiple solutions deliberately but instead gets hacked. For the forwarding parties it is a trade-off between disk space and the possibility of earning through this coin.

A point that could be accounted as a con of this system is that it is not directly possible to hand out change that is risk free in terms of being credit from a trusted node. A goal of the diff process must be to find a best fit for the amount that needs to be payed. Issuing smaller coins will bring less risk to fit any price. However, more coins require more memory space and compare time. Users must decide what is the amount they are willing to give everybody the benefit of the doubt and maybe choose they largest coin value about twice this value.

One interesting thing to explore would be how fast the amount of coin IDs converges to a limited amount while enabling enough liquidity in an economy. The higher the number of issued coins the higher the probability that in the payment process no new coin needs to be issued but an existing one can be used. I would expect the system to settle at a certain amount of coins. The usability of the system would be dependent on how high this amount might be and whether it can be fulfilled with current data storage capacities.

The advantages I see in the systems characteristic:

Verification is based on memory size. Storage is an asset you invest in and then have it. When verification of valid transaction is based on proof of work(energy) it is more like a liability. It costs you for every proof. Further, in this proposal there might be a limit which can be reached by advancing technically, in contrast to bitcoin which endlessly pushes performance without ever reaching a state where it is sufficiently good to fulfill a given task.

Another pro of the system is that the payer does not require internet connection for spending coins. Without the block-double-spending option the system could actually run on a very very slow communication channel. This is base on the fact that information for existing trust lines is distributed in the "saving phase of money". The time between money has been earned until it is spent. There is no need for communication between nodes in the trust line during the moment of payment. For the payment process to happen, only the two interacting nodes need to be active.

Dito makes use of the idea that creation of a coin is cheap. Since it is just digitally created it is fine if a coin is only spent once and then loses its value by informing everybody that this coin should not be accepted any more.

There is high potential for individualism without the need to fork. Every node can manage its own economic contribution. The individual has a lot of possibilities, but also carries a lot of responsibility. It gives the users the freedom to group economically independent of their geographic location or government. Risk management is decentralized and individualized, which is of significance in a monetary system that is not universally backed by tangible reserves.

I have the hope that a system like this is not only designed as decentralized but also stays decentralized during operation. Since again security is primarily based on memory which is easier to distribute (not like mining based on prove-of-work moved where energy is cheap, because transporting energy is not cost effective...)