THE RESILIENCE PROTOCOL

A DISTRIBUTION MECHANISM TO CREATE RESILIENT TRANSACTION NETWORKS

Digital transaction technologies make possible new distribution schemes that allow continous flow of liquidity through all economic participants, essential for demand reactivation.

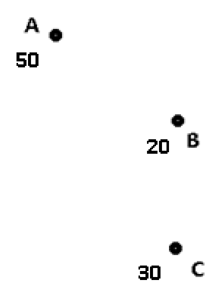
The following is a description of one of such schemes. Inspired in Johan Nygren's resilience protocol, which is based on the weakening of transactional links.

I. Transactional links

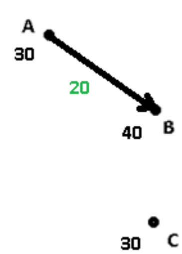
Originally transactional links are an idea of Ryan Fugger's Ripple cryptocurrency from the perspective of credit clearing. As such they work as a guide to get back to an initial state through directional graphs.

With a few diagrams, let's see how they'll work for this protocol:

Say there are three accounts A, B, C with initial balances 50, 20 and 30 respectively.

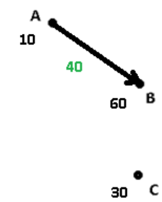


Then say there's a transaction from A to B of 20:

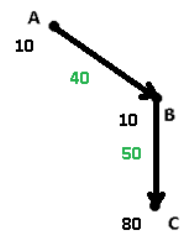


Thus a transactional directional link is created between A and B for the transaction's value (Green). System's balance is still 100: A: 30 + B: 40 + C: 30 = 100. Only now, there's a link that would allow to return to the initial state.

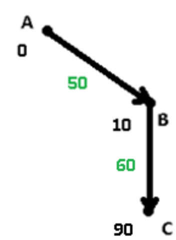
If A makes again a transaction to B the link is reinforced. Say A sends 20 to B again:



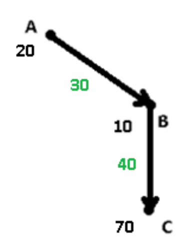
Then say that B makes a transaction to C of 50, creating a new link:



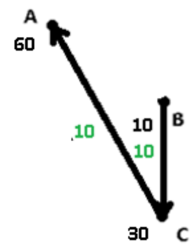
And then, a transaction from A to C of 10, reinforcing each link in the route:



When a transaction is made backwards over a known route, all links of the route are weakened by the value of the transaction. For example, let's say C sends 20 to A:

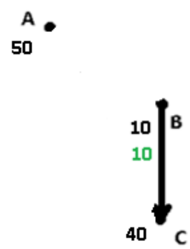


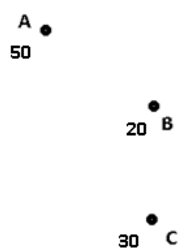
If a transaction is made backwards over a known route, but the transaction is larger than the value of the weakest link, the route is weakened by the value of the weakest link, the weakest link is eliminated (links of value 0 are deleted), and a new direct link is made between sender and receiver for the remainder of the transaction. Say C sends 40 to A:



These rules ensure there won't be loops in the network and there will be only one route between nodes if there is one.

If transactions routes were followed backwards to clear the links, this would act as a guide to the initial state:





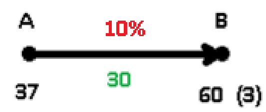
II. Value Tranfer Mechanisms

The network supports its participants by distributing voluntarily added fees. These modify the value of transactional links. First, when a transaction is made, either by over- or under-registering the deducted amount from the sender's balance. This determines the direction of value transfer within the network. And secondly when distributions are made, by weakening links as backtracking transactions that try to clear them.

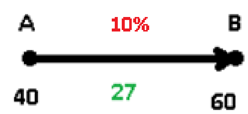
Forward value transfer: deducted amount link fee underregister

First let's see how the mechanism of value transfer works within the network:

Say A sends B 30 at 10%, underregistering the deducted amount 33 as 30 in the link:



The added fee (3) at an auxiliary B's account B' will be later distributed backwards through the link, weakening it:



Note that if the backwards transaction were followed to clear the link, it wouldn't return to the inital state:

Hence, an account's balance is not the difference of incomming vs. outgoing links since there is the value of fees floating about. It's instead calculated from incoming base transactions and outgoing complete transactions, which aren't exactly represented by the links. Instead, links can be can be used to calculate what would be the account's 'value for the network' given by:

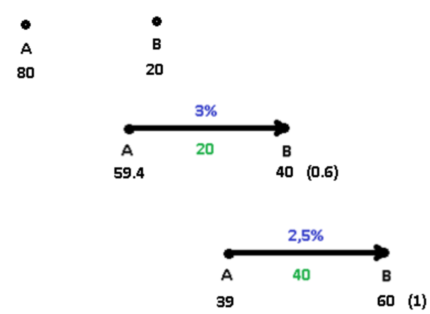
V = B + B' + oL –iL

Where B = 'account's balance' and B' ='account's auxiliary', oL = 'outgoing links', iL = 'incoming links'. Thus, V would be the balance of the account if all transactional links of the network were followed back to 0 and all auxiliaries emptied.

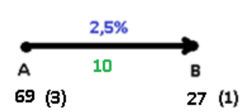
Althought it's one of many design choices, it's recommended that accounts should only receive from distributions when B < V and V > 0. This means the network would try to support each participant up to its value to the network, in other words, the network will try to make B equal to V.

New transactions with different rates alter the sender's link rate by pondering [new rate of link = (previous link value * previous link rate + new transaction's value * new transaction's rate) / sum of previous link value and new transaction's value]. (Note: It only changes the first link in the route).

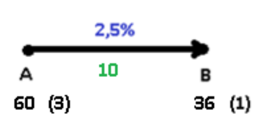
Withholding distributions for this next example, say A makes a transaction of 20 to B at 3%, and a second transaction of 20 to B at 2%:



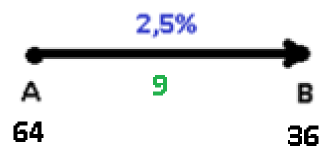
Backwards transactions don't alter the link's rate. (Note: If it overcomes the weakest link's in the route, then it'd be broken and a new direct forward link is made between sender and receiver for the reminder). Say B makes a transaction of 30 at 10% to A:



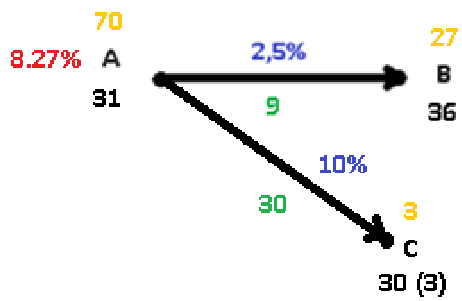
Transactions without added fee (rate 0%) <u>should not be registered in links</u>, this transfers the amount and its complete V. A transaction from A to B of 9 at 0% would be:



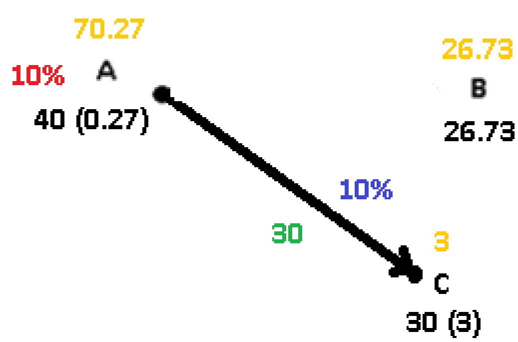
However, **distribution transactions** are registered transactions of rate 0% in links, hence these don't transfer V. Clearing the distributions in this case, dB'A (1) and dA'A (3):   (A' and B' are auxiliary accounts for the fees)



Each account has a rate it makes transactions set by its user, but may also have a **public rate** calculated by pondering the rate of its links PR = Sum(link value *link rate)/Sum(link value). (note: Initial PR is the same as the rate set by user). Say A makes a transaction to C of 30 at 10%:



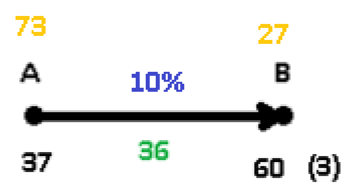
V of each account is displayed in yellow. Let's review the forward value transfer mechanism, say B sends 9 to A at 3% clearing the link. Warning: Clearing links splits the network.



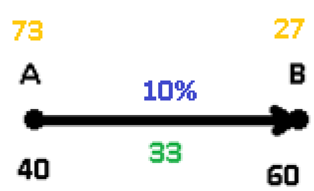
Value transfer direction and amount can be independent of the base transaction's direction and amount. Many different network behaviours and features can be designed around this:

Backwards value transfer: Deducted amount link fee overregister

Let's have A send 30 at 10%, this time overregistering the fee to the deducted amount from A 33:



Checking V for each account, the value transfer can be seen going backwards. Hence, this kind of transaction should be consented by the receiver, its maximum rate should be receiver's public rate, and should be made from accounts with less V than the receiver, again, all these are suggested design options. Distributing backwards we have:



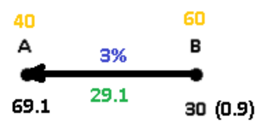
Again, notice that distributions don't transfer value, but contribute to making B = V when B < V

<u>Networks could be designed to transfer value to the party with less value in the transaction, aiming at averaging values across the network</u> (recommended).

Direct value transfer:

So far, it can be seen that value is transferred when links are registered such that backtracking won't take to the initial state. This means, creating links between sender and receiver that don't alter balances in the same proportion transfers network value.

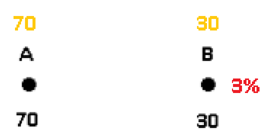
Is possible to make transactions that only transfer network value. Say A sends 30 of NV to B:



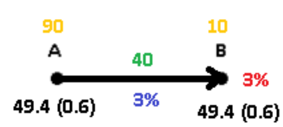
Since this kind of transaction could change receiver's public rate, the rate for these kind of transactions should be of the receiver's public rate. The appeal of this kind of transaction for the receiver would be again, that the network would try to make B = V with its endemic activity.

It's also possible to make mixed forward-direct transactions the design of this feature is open to developer.

Swap balance-value :



Another interesting transfer that could be made between two parties is the direct exchange of balance to value or viceversa. Say A exchanges 20 from its balance for 20 of B's V:



The cost of this Exchange should lie on each participant into the opposite's auxiliary and should be of the bigger rate between both PRs. Note that this operation doesn't require a 1:1 ratio on Exchange, this feature is open to developer.

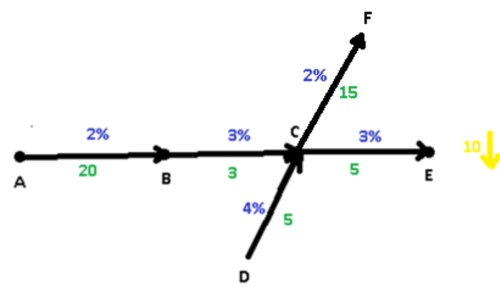
A network could be designed for all accounts to only make forward value transfer transactions. There could be different kinds of participants within the network, some allowed to only make forward value transfer, or only make direct value transfer, etc.

Distribution Mechanisms

Distribution schemes of added fees can be made at developer discretion. Here are some suggested mechanisms.

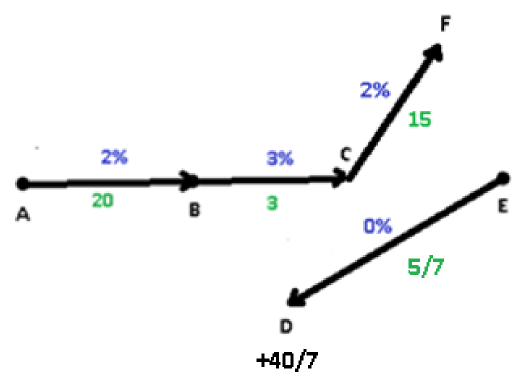
1. **Nygren's**

   The added fee is propagated backwards immediately after the base transaction. Split and pondered at ramifications (what Nygren called 'Dividend Pathways'), until accounts without incoming transactions are found. If 10 were to be distributed from E:

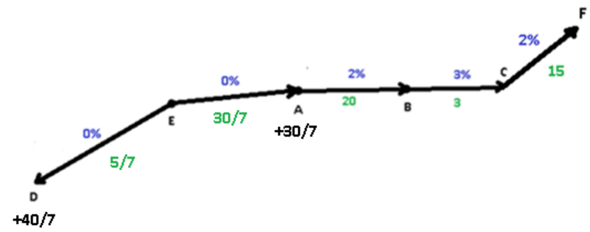


   10 is propagated backwards and splits at C by pondering incoming paths' rates. Through CB goes 30/7 = 10*(3/(3+4)) that goes to A that has no incoming links; Through CD goes the remainder. CF doesn't pull from the distribution since the link is towards F. Thus the distributions are,

   40/7 from E' to D:



And 30/7 from E’ to A:

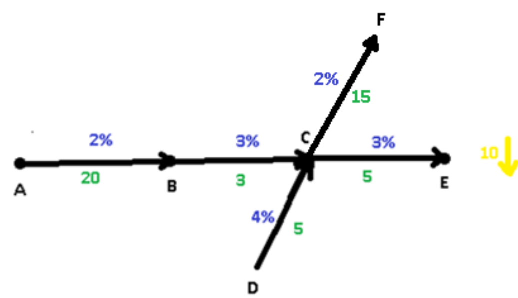


(A different order of these transactions would generate a different yet equivalent graph)

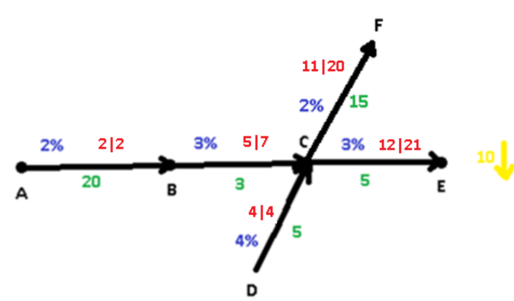
2. **Arcila’s**

The added fee is immediately propagated backwards to each participant pondered by link-fee and distance to the source of distribution:

Let’s say we have a similar situation as before:



To calculate how much each account gets from the distribution, we’ll need the ratio between the first and secondary accumulations of the line rates:



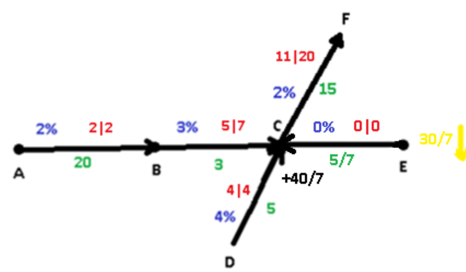
Let's see, first let's check the A – B path. A's an end therefore it's first and second accumulation are the same 2/2, B's first accumulation is A's rate + B's rate (5=2+3) and it's second accumulation would be B's first acc. (5) + A's second acc. (2) hence (5|7).

C is where two paths meet, it can be simplified as having only one path in from the sum of the first accumulated of each path, in this case, it'd be like having (9|9) -> C. This way, CE's first accumulation would be 12 = 9 + 3 and it's second accumulation would be 21 = 12 (CE's first acc.) + 9. CF won't influence the distribution since it's not backwards from E, however let's have its 1°|2°acc. ratio : 11 = 9 +2 First acc. and 20 = 11 + 9 Second acc.

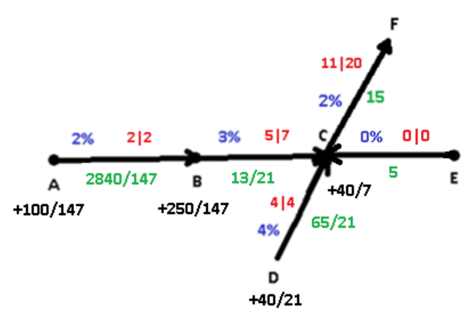
From there, distribution is like this into each account balance:

C gets 40/7 = 10 * 12/21. The remainder 30/7 splits into A-B and D using B's and D's first accumulated: A-B path gets 50/21 = 30/7 * (5/(5+4)) and D gets the rest 40/21. Then A-B path is distributed like this: B gets 250/147 = 50/21 * 5/7 and A the remainder 100/147.

Links are weakened by these distributions and recipient accounts' balances are increased by the distributed amount, just as if they were registered transactions from E (yet at rate 0%). First 40/7 to C:



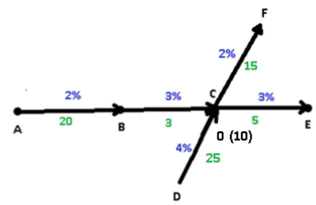
The completed process would be:



All distribution transactions have to be cleared before a new transaction is made.

3. **Angel's (Recommended)**

The added fees are accumulated in auxiliary accounts, and then distributions are made when the number of transactions to the auxiliary accounts are the same number of incoming links + 1. These are made towards incoming links' auxiliaries and self balance by pondering public rates. Say this is the third transaction that C's auxiliary account receives and has accumulated up to 10:

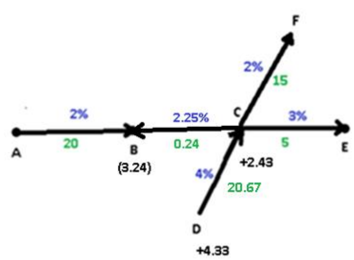


C's V is 2 and B is 0, so it participates in the distribution (B<V)

C's public rate is 2.25 = (2*15+3*5)/(15+5). Distribution is made as follows:

- C receives 2.43 = 10 * (2.25/(2.25+3+4))
- B' receives 3.24 = 10 * (3/(2.25+3+4))
- D' receives 4.33 = 10 * (4/(2.25+3+4))

Note that D' meets the distribution condition so D' distributes to D. All cascading distributions have to be cleared before a new transaction is made. This distribution also weakens DC link and changes BC link's direction, inheriting C's public rate:



Note that weakening links could eventually clear them. This could pose a challenge for network cohesion.

A feature open to developer's design would be self-distribution, taking a part of the amount to distribute directly into the balance of distribution emmiter.