Newrl – A Scalable, Compliant and Inclusive Trust Infrastructure

Executive Summary

Much of the innovation in public blockchain technology has remained focussed on the narrow domain of crypto-native assets. Mainstream use-cases continue to be hard to implement on current public blockchains. These include loans to small businesses or thin-file borrowers, tokenization of contracts and assets like property, stocks, invoices, brands etc and active collaborations that require identifiable persons.

This is partly driven by lack of on-chain identity and inadequate legal enforceability of on-chain transactions. Both these limitations are also the central tenets of most of the current public blockchains (i.e. they are "features, not bugs"). Mainstreaming of blockchains is hard also because current chains are monolithic and difficult to scale, despite several innovations in sharding, layer-2 and inter-chain bridges. Lastly, most of the current public blockchains remain energy inefficient, opaque in governance and open to creeping centralization.

Newrl (pronounced "neural") is a layer-1 blockchain aimed at addressing the needs of the real-world use-cases - such as borrowing and lending, tokenization of assets and easy collaboration amongst people. It is a "trust network" that uses continuous trust assessment of its participants built from their own on-chain behaviour as well as their community's inputs about them. Newrl's protocol is based on identity at the chain layer and legally robust tokenization. Also, to take blockchain technology beyond the highly skilled but relatively small web3 developer community, Newrl employs simple-to-use, low-cost and highly scalable smart contracts and decentralized autonomous organizations. These are at-node no-code templates which are parametrized on-chain – analogous to an "app-store" approach to smart contracts and DAOs.

To enable scalability with decentralization, its consensus protocol uses trust scores of participants and a human-brain-inspired architecture for attention prioritization. Newrl is also created to enable easy side-chains and bridges - so that instead of a monolithic architecture, it promotes diversity of specialized-use sub-networks.

Newrl is built as the blockchain protocol for the real-world applications. It is designed as a foundation for a scalable, compliant and inclusive web3.

Contents

Introduction

Newrl is a 'Trust Network' – a layer-1 blockchain built for mainstream decentralized finance. It enables individuals and small businesses to access capital from their communities using their credibility and tokenized assets as collateral. To this end, Newrl simplifies legally robust tokenization of assets like stocks, properties, and start-up equity. It also enables individuals and small businesses to tokenize new forms of assets like personal creditworthiness, social media revenue inflows, patents, invoices, brands, warehouse receipts etc. For illiquid assets, Newrl supports a mutualization mechanism to facilitate liquidity, also helping their use as a collateral in a loan.

An immediate embodiment of decentralized social finance is 'microbanks'. These are community-driven cooperative lending units set up as decentralized organizations on blockchain. Starting with intra-community lending, microbanks can invite outside institutional pools of debt capital through frictionless on-chain securitization. A community in this context can be any group of persons with a similar affinity - be it residents of a town, employees of a company, immigrants from a country, alumni of a college, enthusiasts of a multi-player game, fans of a movie franchise etc.

The central unique feature of Newrl is the trust scores of its participants - which are influenced by both the community and on-chain transaction history. Other unique features of Newrl include confirmed-identity participants through decentralized KYC, legally robust tokenization protocols to convert real-world assets into on-chain tokens and the use of ready templates for smart contracts as well as decentralised organisations.

Newrl employs a novel consensus protocol called 'proof of trust'. Individual nodes are run by real persons with clear KYC and a presence on the trust network. These nodes participate in validating transactions and adding blocks - earning governance tokens for good contributions as well as a higher probability of future selection for block addition. Conversely, they lose their deposited tokens for bad contributions or malicious behavior and also get a sharp reduction in probability of future selection for block addition.

The thesis behind Newrl is that there exists a moderate amount of trust amongst individuals and small businesses that can be harnessed for efficient commercial transactions amongst them - something that is already common in most real-life transactions. The current public blockchains focus on purely trustless transactions amongst anonymous participants. This misses out on the benefits of the informal non-zero trust within a community. Newrl enables explicit recognition and tracking of this trust amongst its participants. To be clear, Newrl's transactions and validation protocols are as robust as those of current public blockchains and do not require mutual trust between transacting parties. The trust scores are additional features available to the participants to use.

For scalability, Newrl's computations are carried out in a manner inspired by the human brain (hence the name!). Like the organic neural circuits, Newrl's computation and storage is robust and yet energy efficient. The main idea is that decentralisation does not mean endless over-redundancy - something common to most of the current public blockchains. Newrl's architecture is based on distributed storage and computation - which achieves decentralization without repeating all storage and computations as many times as there are nodes in the network. The aim is to keep the actual cost of computation and storage within the same order of magnitude as the costs (actual, not the higher ones charged to customers) of existing centralized systems such as payment rails like Visa and Mastercard as well as bank transfers.

Background: Limitations of current public blockchains that we are looking to address

The Bitcoin project[1] ushered in a remarkable era of decentralized handling of value over the internet. It had a strong anti-establishment bias and was proposed as an alternative to fiat currencies for peer-to-peer payments. This distracted most lay observers from the revolutionary potential of the decentralization technology itself, as debates around the projects focussed a lot more on the economics of fiat currencies and their alternatives.

However, the second generation blockchains - starting with Ethereum[2] - broke free of the payments-only objective and also diluted the strong aversion to fiat currencies. Instead they focussed on decentralized handling of generalized transactions including but not limited to payments. This was driven by use of smart contracts which are autonomous algorithms that execute a predetermined set of steps in response to specific inputs from the users. Smart contracts allowed issuance of tokens on a blockchain beyond its primary currency (e.g. ether on Ethereum). This was a promising start to enable uses of public blockchain in mainstream finance - capital markets, banking, insurance, remittances and so on.

However, as various start-ups and established firms tried to implement specific use-cases from non-crypto-native domains in the second generation blockchains, they came across several problems. Some of the major ones are as below.

1. Handling non-crypto-native assets (or so-called "real world assets") in a legally robust manner is difficult in current public blockchains. These platforms are technologically advanced but legally primitive[3]. The most persistent of these issues for non-native assets is the problem of double tokenization. There are no solutions to this in the current frameworks, partly due to

the aversion - built into existing blockchain projects' ideology - to the current judicial systems run by governments.

2. Identity is an afterthought in current frameworks. In non-crypto, real-world transactions, this is difficult to work with, least of all due to existing regulations on know-your-customer and anti-money-laundering when it comes to typical assets and loans.
3. Most of the current public blockchains are overly focussed on trustless transaction handling. In the mainstream economy, there is varying level of trust between parties - ranging from near-zero to near-perfect[4]. This partial trust is not harnessed in the existing frameworks.
4. Current public blockchains do not have memory of historical behaviour of participants owing to their focus on anonymity. That misses out a very useful input to assessment of individual trustworthiness.
5. The current blockchain infrastructure is hard to scale. There is excessive redundancy of computations and storage - well beyond what is needed for sufficiently decentralized governance.
6. Smart contracts in current public blockchains are expensive. A very simple contract on ethereum may cost as much as $500 to deploy while a complex contract can be over $10,000[6]. They are also prone to bugs and hacks. Complex contracts - of the type that are likely to be useful in real life - are hard to create and expensive to deploy. Some of them are even infeasible given the upper limit on contract size byte code.
7. The mining operations are run opaquely by specialist nodes. In recent times, they are known to routinely pool resources[7] - thus effectively centralising the chain operations without many of its users even realizing it. In general, there is a chasm between those that maintain the network without really using it and those that actually use it for something but do not participate in maintaining it.
8. The control over the blockchain protocol is not decentralized for most of the current public blockchains. While some projects are spread out in terms of control over their codebase, the 'community' managing it does so without formal voting mechanisms. Several other projects are centrally controlled. Of course, a few good exceptions like MakerDAO do exist.
9. Current public blockchains are built with one currency for each chain - leading to concentration of influence. There is also an obvious conflict of interest in reducing the cost of transactions on the blockchain because it may reduce the value of the underlying currency in which this cost is paid.

Some of the above problems are based on the legacy of current projects in the liberatarian/anarchist anti-establishment ethos that was and is widely shared by early adopters of bitcoin. Hence many issues like lack of identity and presence of a platform currency are not bugs but features of these projects. Some of the other problems, while solvable, require a lot of tweaks.

There is a strong case for a mainstream-focussed blockchain to begin afresh by incorporating a lot of the good innovations from the present state of the art in the crypto-native domain while squarely addressing the above limitations at the design level itself. This approach is likely to be more fruitful in the medium term than trying to introduce ‘patches’ to existing blockchain projects.

Newrl as a blockchain for mainstream finance

The limitations mentioned in the previous section are why Newrl was conceptualized as a fresh start. Besides a novel consensus protocol named ‘Proof of Trust’, there are several features unique to Newrl that enable it to address these limitations of existing blockchains.

a. Identity at protocol layer
b. Trust network
c. Legally robust tokenization
d. No-code templatized smart contracts and distributed organizations
e. Neural architecture of Newrl
f. Mutualization of tokens
g. Transparent protocol governance

We describe the above features in greater detail in this section. The ‘Proof of Trust’ consensus protocol is described in the next section.

<u>Identity at protocol layer</u>

Each wallet on Newrl is linked to specific documents that prove the identity of a person - be it a natural person or artificial (e.g. a company). The documents themselves are maintained either locally at the user’s end or with an institutional record-keeper based on the user's preferences. The hash of the document number as well as the document’s digital copy is stored on-chain.

In addition to the hashes of identity-proving documents, each wallet also incorporates information about jurisdiction of the user and other attributes like the user being an accredited investor. Such information is useful in compliance with regulations around the underlying assets of tokens.

Newrl’s transaction validation protocols require that the public address used for creating them belong to a ‘KYC-ed’ wallet. The full list of these wallets is stored in the state database. To be included in this list, a user has to create a new wallet by submitting hashes of the KYC documents as mentioned above while storing the documents themselves with an institutional record-keeper or locally. In either case, the addition of a new wallet to the state database is carried out through an explicit transaction (labelled 'type 1') on the blockchain. This transaction is signed by the institutional record-keeper or one of the existing users with a valid wallet (someone that vouches for the identity of the new person) depending on the choice of the user for document storage.

All activities on Newrl involve the use of such KYC-compliant, verified identity wallets. This includes economic transactions as well as validation and block creation.

The identity portion of Newrl, for individuals, can be further generalized in future to make it based on entirely decentralized identification whereby an individual stores iris and fingerprint information locally on her device and submits the hashes of these to the chain. Additionally, individuals vouch for each other's identity, making the whole trust network robust over time. As this becomes mainstream, reliance on government-provided identity documents will wane. Artificial persons like companies can be identified through their creators' identities.

Trust Network

The trust network is a system to create and maintain a network of peer-to-peer scores of trustworthiness based on user-defined values and transaction-based updates to them. In the abstract, the trust network consists of the following components.

1. Persons:

   Each person is identified by a personId. One person can have the maximum of one personId in Newrl. However, it can be mapped to multiple wallets. This is akin to a person having a single tax id but multiple bank accounts all mapped to it.
2. Connections between pairs of persons:

   A connection is defined as a link with exactly one source person and one destination person. Connections are unidirectional.
3. The trust score of a connection:

   Each connection has a trust score which is a real number between -1.0 and 3.0. The trust score indicates the assessment of trustworthiness of the destination person in the eyes of the source person.
4. Transactions:

   Each person can carry out a transaction that may involve one or more of the other persons. A transaction may also have only one person involved in it.
5. History:

As a blockchain, Newrl has a full history of transactions. This can be used by the participants as they deem fit to derive a summary score of the participant's behaviour. The network itself does not have specific algorithms to convert the history into a history-based score.

6. Mechanisms to modify the trust score of a connection:

   The trust scores for each connection can be modified by the source person of that connection. It also gets refined with each transaction that involves the two persons. Also, inaction like non-adherence to a contractual obligation by a person alters the trust score. The source person can always reset the value of trust score for a connection, overriding any automatic updates it might have had.

7. Network integrity maintenance by verification protocols:

   On Newrl, anyone can report suspicious activity by a specific person on the network. Upon such a request, specifically designated auditor nodes can verify the accuracy or lack thereof of such reports. If a person is verified to be engaged in malicious activity, his/her/its personId is included in the grey list table in the state of the network, available to all participants to view, and network trust score (described below) of that person is reduced to -1.

The trust scores can be queried by any participant in Newrl. For instance, in the use case of borrower assessment in a lending transaction between persons who do not already have a high-trust relationship, the lender can check the distance on the trust network between the borrower and herself and also the trust scores of the outward connections from the lender to the borrower. The lender can also check the historical behavior of the borrower in previous transactions - lending or otherwise.

Trust network schematic is described below in diagram 1.

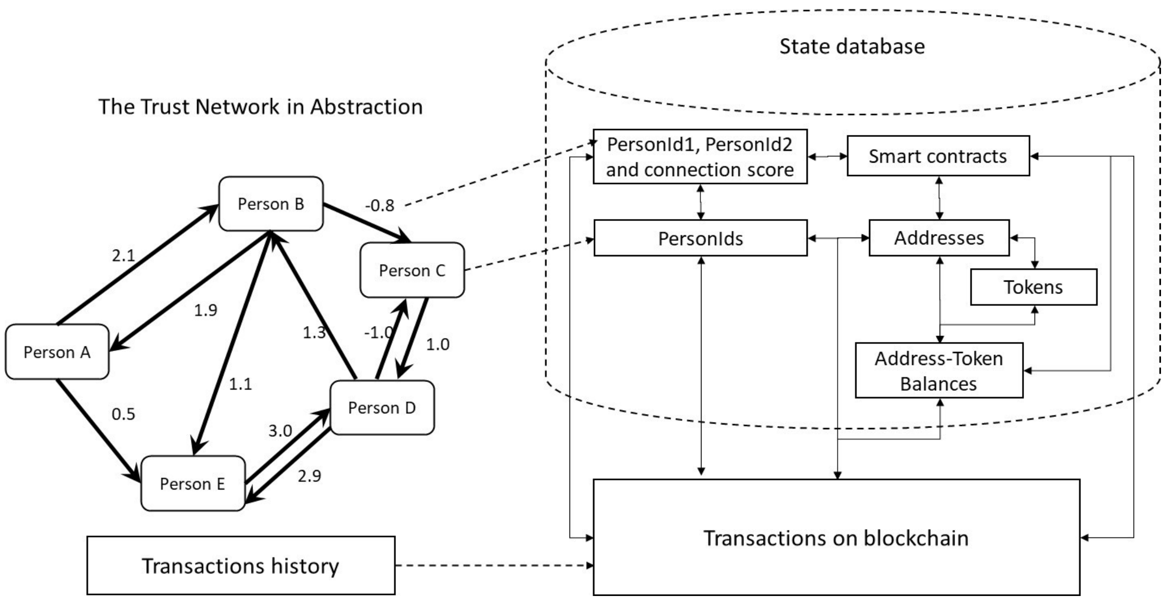

Diagram1 : The trust network in the abstract and concrete

Since Newrl is a public blockchain, we expect several alternative algorithms to evolve that will use the trust score and transaction history information. Other use-cases involve counterparty risk assessment in a contract between two small businesses, evaluation of an entrepreneur by angel investors, a real estate transaction escrow etc.

Overall, the trust network can be used in three different ways.

A. Peer to peer assessment of one person on the network by another for a proposed new transaction, based on the multiple paths connecting the two persons.
B. General trustworthiness assessment of a person by another person without reference to specific path connecting the two on the network.
C. Assessment of a person based only on the history of transactions carried out by them.

Appendix 2 describes the use of summary trust score using an algorithm similar to Google's PageRank.

Legally robust tokenization

Unlike the current public blockchains, token issuance in Newrl is not subsumed under smart contracts. Also there is no hierarchy of tokens where the native currency of the blockchain is the primary unit of account. All tokens have the same importance on Newrl. The balances of tokens in wallets are maintained in the state database in Newrl, unlike the current blockchains where token-specific data is stored inside its smart contract, making it prone to bugs and hacks.

Token issuance itself is a standard transaction type (labelled 'type 2'). It is signed by a "custodian" - which refers to the actual asset keeper in case of custody-based tokenization and creator of the asset in case of native issuance based tokenization (see below for further details). There is a 'first owner' of a token in whose wallet the newly created tokens are added. Upon successful inclusion of a token creation transaction in a block, the state database is updated for the new token and the balance of the first owner's wallet is updated with the new token.

To make the tokenization legally robust, Newrl includes two-way mapping of underlying contracts to tokens. To link the tokens or smart contracts on blockchain with a typical legal contract outside the blockchain, the executed legal contract document is cryptographically hashed. Next, this hash is included in the transaction submitted to the blockchain network for validation and inclusion in the next block. In the legal document itself, the blockchain is identified through its genesis block hash, one of the recent block hashes and the token creation transaction itself through its identifier on the blockchain. This is described in diagram 2 below.

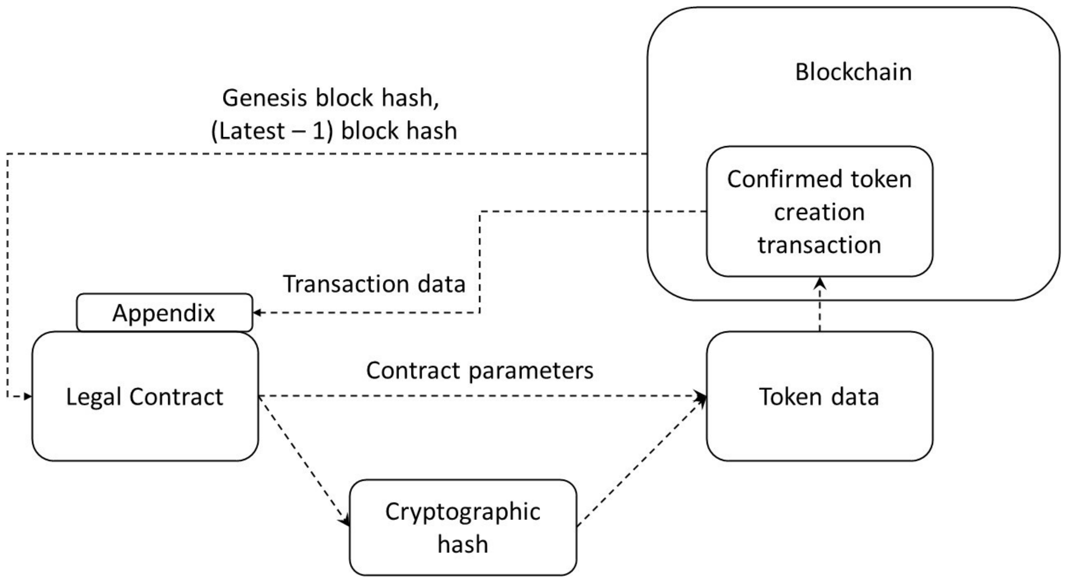

Diagram 2: Two-way mapping of an off-chain contract and on-chain token

Assets as well as cash-flow-bearing contracts can be tokenized[8]. Assets can be tokenized through either custody-based issuance or native issuance. Contracts can be tokenized through native issuance.

a. Asset tokenization through custody

In this mode, the asset - financial or real - is transferred by an owner to a pre-identified asset-keeper. The asset-keeper and the owner execute a contract that transfers the beneficial ownership and interest in that asset back to the owner from the asset-keeper in the form of tokens created on a blockchain. The ownership of tokens is based on the wallet they belong to on the blockchain. The tokens are transferable. They can also be converted back into the asset (i.e. de-tokenized) by their owner. This is represented in the diagram 3 below.

b. Asset tokenization through native issuance

Native issuance refers to the tokens created being the sole representation of the asset. It works in instances where the issuer/creator of the asset is involved in the tokenization process. The issuer/creator moves its ownership records of the asset (e.g. a share register) to the blockchain and creates transferrable new tokens that refer to the asset issuance contract (e.g. shareholding agreement or bond issue document) through the two-way mapping process described above. In case of native issuance, since the tokens directly represent the asset, there is no custodian or asset-keeper involved.

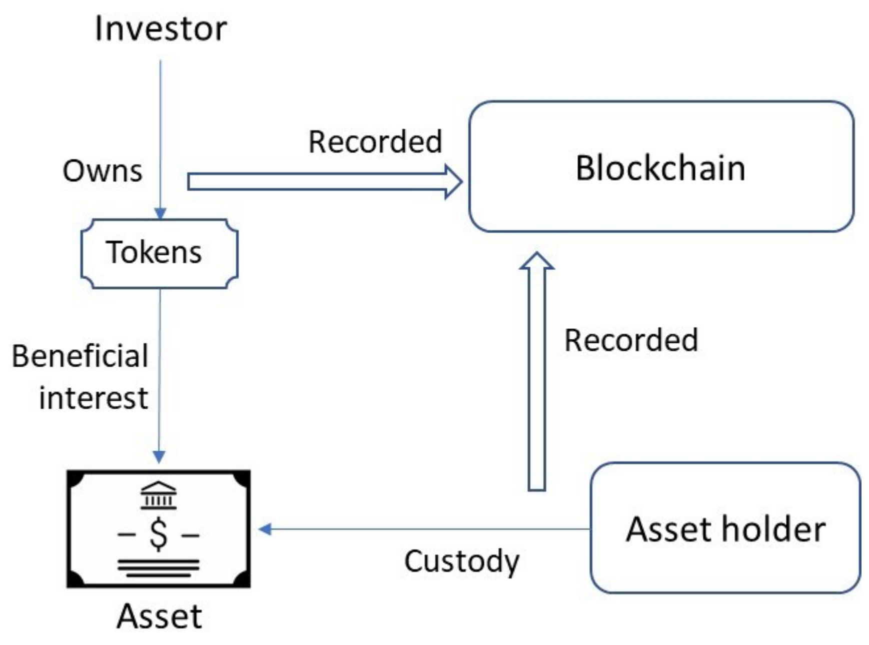

Diagram 3: Asset tokenization through custody

c. Contract tokenization through native issuance

Contracts like loans, derivatives and revenue sharing arrangements are tokenized by creating a new token mapped to the specific executed contract, as mentioned above. In case of contracts, it is important to note that the underlying conditions of the contract should also be translated into smart contracts on the blockchain platform, to the extent they are deterministic. An important reason to tokenize a contract is to enable it to be transferred. This part too needs to be in line with the underlying contract's specifications about the feasibility of transferability of participation in that contract.

A more detailed description of tokenization is covered in a separate white paper.

No-Code smart contracts and DAOs

Given the focus of Newrl to be scalable for mainstream finance use-cases, it uses node-level templates for smart contracts and decentralized organizations. In this approach, instead of burdening the chain with the full bytecode of each contract, only parameters are stored on-chain while templates are drawn from codebase in local nodes. This approach gains in scalability and robustness by giving up the high (and typically quite underutilized) versatility of an all-purpose 2nd generation blockchain like ethereum.

Smart contract and DO templates are created by the developer community and each node has the full codebase for the templates. The idea behind templatizing is that more than 90% of use-cases, especially within a specific industry like finance, are covered by just 1% contract types.

The contract super-types and types at the beginning are as below.

1. Loans - secured/unsecured, bullet repayment/spread out repayment.
2. Aggregation and splitting - many contributors to one pool, one inflow split into many recipients; useful for creating pools/funds from individual assets.
3. Tranching of risk - waterfall of payment preferences; useful for securitization of loans.
4. Mutualization - described above.

Smart contracts can be combined as well as nested. For example, one set of persons can join hands to aggregate their tokens and then mutualize the pooled tokens with another aggregated contract's tokens.

Smart contracts follow a three-step process for creation and deployment.

a. The creator of smart contract instantiates a specific contract by specifying a template using a transaction of 'type 3'. This creates 'child transactions' which may include transfers, wallet creation, token creation etc.
b. The participants in the contract sign the child transactions and submit them to the network. These are confirmed upon additions to blocks or rejected if not valid.
c. If the prescribed minimum number (which can be all) of child transactions are confirmed, the creator triggers deployment of the contract. This makes the contract 'live' and from this point onwards its code executes upon triggers based on calls by other transactions or time.

The decentralized organization templates included in Newrl protocol at the beginning are as follows.

1. Membership DAO - enables confirmation of membership of a group using rules set by the community.
2. Microbank - enables pooling of money tokens to lend and tranching of risk of loans. Described further in appendix 3.
3. Mutualized insurance - enables pooling of money tokens as premia received from insured persons and pays out to specific claims upon insurable events having occurred.
4. Decentralized exchanges - enables trading of specific tokens in a decentralized manner.
5. Venture funds - enables pooling of money tokens to fund new projects in exchange for equity ownership in them.
6. Mutual funds - enables pooling of money tokens to invest in listed securities.

7. On-chain firm - enables creation of a typical business vehicle with founders, management, employees and passive investors. Allows diverse types of ownership in the full spectrum between employee and investor. Includes one-person firms that enable artists to raise funding for specific projects as well as as a general investment in their work.
8. On-chain non-profit organization - enables creation of a non-profit business that accepts contributions and deploys them for specific non-profit purposes (philanthropy, academic research, social sector projects etc).

When a user instantiates a DAO template on Newrl, they create a DO wallet and trigger specific child transactions relevant to the DO's deployment. This is similar to the smart contract deployment above. The DO templates differ in that they allow a wider leeway for changes in the contracts governing the DO. The primary thrust of a DO is to enable multi-user governance. As such, the DO template includes a variety of multi-signature rules for its main wallet and sub-wallets. Overall, a DO is expected to be a persistent collective of human users that have a transient set of rules governing it.

Neural architecture of Newrl

Architecture of Newrl is inspired by the human brain - hence the name! The brain is a system based on distributed computing and storage[9]. It is plastic and robust to considerable damage to one or more of its parts. It is also energy efficient while still being robust in data retention and processing reliability. This is made possible by the generalized design of individual subsystems carrying out specific functions and coordinating with other subsystems for creating a coherent outcome. Also, there is a dynamically shifting 'attention' – which activates only specific parts of the brain at a point of time and not the whole brain. This attention is proportional to the relevance of external stimuli to the wellbeing of the organism.

Newrl is designed along similar principles, described below.

1. Each node on the network stores only part of the data required for the operation of the blockchain.
2. Each node participates in only part of the computation during the operations.
3. All transactions and all blocks are not equal. Some are more valuable than others. The network spends more resources on larger-value transactions and blocks than smaller-value transactions.

The reduced participation mentioned in 1 and 2 above is managed through random selection of block-creating sub-group of nodes as well as random selection of transaction validators. Point 3 above is implemented through reducing or increasing the number of block creators as well as number of transaction validators based on the importance accorded to the transaction by its proposers – through the transaction fees. The degree of ‘distributedness’ is a function of both the size of the state and chain as well as the size of the network. At the beginning, when the network is small and so is the blockchain, this architecture defaults to full redundancy across all nodes. However, as the size of the chain increases, distribution of storage and computations kick in.

This allows Newrl to be highly scalable right from the beginning. Distributed nature of its storage and computation keeps the real costs of transactions on Newrl within 1 order of magnitude of the costs of centrally managed ledgers like payment systems. A detailed description of this architecture is covered in a separate white paper.

Mutualization of tokens

Tokenization by itself has limited utility if there is no liquidity. To this end, Newrl enables mutualization of individual tokens into supertokens that greatly expand the pool of interested persons in trading it. This enables owners of individual tokens to access liquidity for their own tokens by converting them into mutualized tokens. It also gives greater monetizability to individual tokens in their use as collateral, because lenders can mutualize a token to sell it while enforcing a loan repayment.

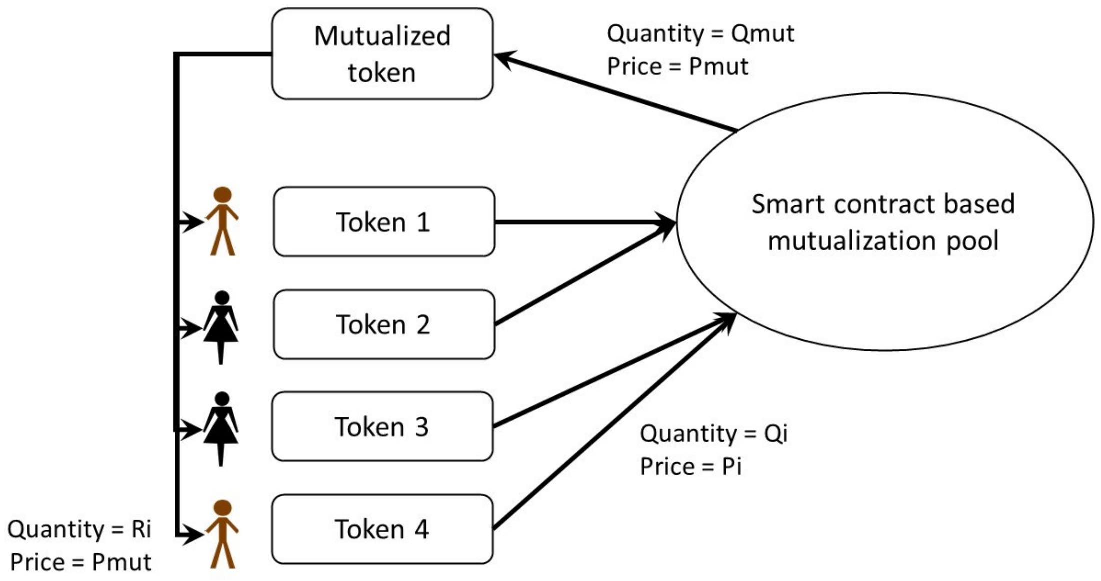

Diagram 4: Mutualization of illiquid asset tokens

Mutualization also helps reduce risk. The idea behind mutualization is pooling of individual assets to create a pool without having to use money transfer for the transaction. The mutualization process is described in detail in appendix 4. Diagram 4 above shows the same pictorially.

Transparent protocol governance

Newrl's governance is managed by the community through its governance tokens. The code repository is in the public domain. Specific changes to it can be suggested by anyone through pull requests. The governance token owners vote on proposed changes and the accepted ones are implemented by ASQI - acting as the custodian of the code. Over time, such code changes will be automated by linking them to smart contracts on Newrl controlled by the governance tokens directly. This will remove even the limited dependence on the custodian of the code. However, even in the current version, the changes continue to be transparent to the public.

"Proof of Trust" Consensus Protocol (PoT)

There are three types of computations carried out in a typical public blockchain[10].

1. Transaction validation when transactions are submitted by individual nodes to the network
2. Block creation using validated transactions
3. Verification of a block upon its broadcast and reception by other nodes, and subsequent update of local state and chain for valid blocks

Transaction handling

In PoT, Any node can broadcast a transaction to its peers. Upon receiving a transaction, the receiving node validates the transaction and if found valid, broadcasts it onwards to a specific number of peers only. This number is a function of network size and transaction fee. For a small network, transactions are sent to all peers. Further details are covered in appendix 1. Receiving peers ignore transactions that they already have (checked using transaction id). All valid transactions are stored in the local memory pool by each peer.

Block creation

PoT involves block creation by a chosen node and verified by a randomly selected committee. There is no competition for adding a new block and no proof of work required to agree on a new block addition. Instead, for arriving at consensus, the network undertakes the following steps in each period of time for block creation.

1. Each node executes a “validation nodes selection program” to select the minting node and the committee, where the selection probability is proportional to the network contribution trust scores of the live nodes.
2. The node minting a new block incorporates as many transactions in its local memory pool as fit a single block - ordered in terms of their timestamp.
3. The node also runs the ‘network trust score updater’ algorithm using receipts from the previous block creation process.
4. The selected node mints the block and broadcasts it to others in the committee. The broadcast includes the block data and signature of the minting node.
5. The selected node also updates its local state. The transaction fees from all included transactions in the block are transferred to the treasury address of Newrl. The treasury pays out the pool periodically and proportionately to the owners of Newrl governance tokens.
6. The others in the committee verify the block using the block validation algorithm.
7. If the added block is not signed by the expected node selected for minting, they ignore it and wait for the correctly signed block. If the correctly signed block received by them is valid, the committee members append it to their local copies of the chain, broadcast to other committee members signed receipts.
8. If the received block is signed correctly but is invalid, the committee members do not append the block to their local chain and just send their own signed receipts with the block index, block hash and their “-1” vote.
9. Other members receive the receipts and include them in their own memory pool upon validating them.
10. At this point, each node in the committee calculates the score-weighted probability of a block being valid, using the available receipts, with receipts from more trusted nodes weighing more than those from less trusted ones.
11. If the block is valid, the committee members individually broadcast the block along with their local signed receipts of the committee members (including their own) to their peers in the network.
12. If the added block is invalid, the committee members ignore it and instead individually mint an empty block with the timestamp equal to that of the previous block plus 1 second and broadcast their list of receipts along with the empty block.
13. In case of non-live minting node or inadequate quorum, the committee members add an empty block after waiting for a stipulated time for a new block.

Each node is also required to deposit a fixed number of valuable tokens of any type in a dedicated smart contract that governs the protocol. In addition to reducing the trust score of a node that produces erroneous blocks, its deposit in the smart contract is reduced and added to the deposit of a specific audit cost pool address.

<u>Repercussions of voting</u>

Each process of validation of a block results in a vote by a committee member of the block being valid or not. These receipts are broadcast at first by their creator and then onwards like standard transactions by the others in the network. At the time of new block addition, the minting node incorporates as many receipts as possible, with preference to older receipts, inside a special "network trust score update" transaction.

The network score update algorithm uses the receipts with a specific logic that rewards honest nodes and punishes dishonest nodes based on receipt vote and actual status of the block.

A detailed description of the PoT protocol is in appendix 1.

Use-cases

Since Newrl is aimed squarely at mainstream finance, it is useful to look at specific use-cases that are enabled by it. While some of these can be implemented in the current public blockchains, the issues related to anonymity, lack of robust legal basis and scalability continue to remain as significant challenges, besides the full list described earlier in this document.

<u>Microbanks</u>

The common limitation of a peer-to-peer lending system is that the pool of individual/small business borrowers is large but that of individual lenders is limited. Institutional debt capital is typically easier to tap into but is constrained by limited trust in the loans. A microbank removes this friction by making the community as the primary lenders, underwriters and junior lenders in a loan, while institutional debt capital is secondary and senior lender. This achieves several objectives, difficult to meet in a purely "lend-to-a-stranger" model of conventional p2p platforms.

- The primary lenders know the borrower and thus can tap into their informal but reliable assessment of the latter.
- The primary lenders also act as junior lenders in the tranched securitized cash flows of the loans. This makes the senior lenders believe in their assessment even more.

- Both primary and secondary lenders are exposed to a pool of loans instead of a specific loan.
- The lenders collectively control the sign-off on each borrower. This utilizes the wisdom of the crowd and reduces possibilities of fraud.
- The borrowers have added incentive to not wilfully default because they know the lenders. It is not an unknown stranger that is affected by their default.

Microbank as a decentralized organization template in Newrl allows any group of persons to come together and start to lend amongst themselves. The trust network allows easy coordination amongst persons as they form the group. While every person does not know every other person well, they are brought together through a few strong ties of high trust amongst several pairs of them.

Use of community's assessment of a borrower can significantly improve credit access to currently underbanked segments[11]. A more detailed microbank architecture is described in appendix 3.

Access, liquidity, and monetizability in real estate

Consider the case of property tokenization by several individuals. Each of these tokens are based on specific houses which are unique even if similar. Hence for an outside investor, the specificity of each property and its token is likely to be a hindrance in buying the tokens. The resources required for assessment of each token may not justify investing small amounts in them. Hence, despite having fractionalized the property, the tokens may not find adequate interest from outsider investors.

If these property-owners were to mutualize their property tokens, the mutualized token starts to represent a specific neighbourhood, a city or even a state, depending on the number and diversity of contributors. For a passive outside investor, this is a much easier assessment to make. The diversification of individual risk achieved by mutualization greatly helps reduce the need for single property assessment.

An important implication of mutualization is the reduction in the number of individual tokens that need to be liquid. In absence of mutualization, if N tokens are created, all N need to be individually liquid. However, if these N tokens are mutualized, the output is tokens of only one type. The bare minimum number of interested parties for trading in this one type of mutualized token is N (i.e. the contributors). If each property had on an average K interested parties in it, the interested investor pool now expands to N*K. This can be adjusted for overlaps in interest by introducing an 'overlap factor p' (with value between 0 and 1) thus making the number of interested parties N*K*(1-p).

The increased outside interest in purely passive investment in a specific class of properties will add to this pool, which was otherwise not interested in individual property tokens. This takes the total pool size to N*K*(1-p) + O where O is a function of N itself as well as the quality of the pool (combination of similarity in theme and diversity in idiosyncratic risk).

The overall increase in interested parties is from 1+K to N*K(1-p)+O. Given the non-linear nature of liquidity growth in asset markets, this can significantly improve tradability of the mutualized tokens, thus benefiting all the property owners as well as the investors. This is described further in appendix 4.

Angel Fund DAO Using Unlisted Equity Tokenization

The case for mutualized start-up equity is similar. In this context, the interested investor pool in each start-up's equity includes the founder, angel investors and employees. Even start-ups in a given sector mutualize their equity tokens, the individual contributors benefit from gaining exposure to the overall sector without being too reliant on a single company's fortunes. Similar to the real estate example, this can also attract outside capital which would have otherwise stayed out owing to lack of bandwidth in evaluating each individual company.

The supertoken can also be thought of as an angel fund, set up as a DAO. Such a DAO can issue more ownership tokens to raise money to invest in its constituent companies as well as new companies similar in theme. The DAO can manage the investment decisions.

The value of employee stock options is a lot more immediate for the employees because they can contribute their shares to the mutualized pool and get a lot more liquid 'supertoken'. The very availability of this option also prompts easy monetizability of the individual tokens as well as mutualized token through their use as collateral in a loan because the lender is aware of the ease with which the token can be sold - directly or post-mutualization. This is described further in a separate paper.

Community coin through mutualization of personal guarantee

IOUs issued by individuals can be mutualized to create a common currency amongst them. Typically, a person would accept an IOU token from another person either in the context of the latter borrowing from the former or postponing a payment to the former. It is not common to use IOUs to fully settle payment obligations. However, if a sufficiently large group of individuals mutualizes its IOUs, they can start accepting the mutualized collective IOU for settling payment obligations.

This is tantamount to a fiat currency issued by the collective of individuals. It is backed by their collective creditworthiness as well as willingness to accept it as payment (for goods and services as well as assets). If the collective creditworthiness of the participants is high enough, outside users can also choose to use the mutualized IOU for payments involving the participants i.e. to pay them or be paid by them. Furthermore, for a mutualized IOU backed by a sufficiently large number of participants of adequately high creditworthiness, the outside users may even choose to use it for payments amongst themselves i.e. where neither party to a payment is one of the participants of the mutualization pool.

The mutualization process can also be stacked. Hence multiple mutualized IOUs can themselves be mutualized thus pooling an even larger number of participants. The logical conclusion of this process would be a global fiat currency that is a mutualized IOU of all participants in the global economy. Short of and in addition to this final step, there can be several city-level, state-level mutualized IOUs. Also, non-geographic collectives of individuals may choose to mutualize their IOUs based on common interest and/or background (e.g. alumni of an illustrious institute).

The immediate relevance of mutualization of IOUs is to enable groups of people to create money for suitable end-uses without having to rely exclusively on the banking system. In the current financial system, only banks can create new money based on their lending and as allowed by regulations like capital adequacy. This system is inherently inegalitarian and is often quite discriminatory. By enabling individuals to tap into their collective creditworthiness, mutualized IOUs put the power to improve their economic wellbeing in their own hands.

The long term relevance of such mutualization is to decentralize the conduct of fiscal and monetary policy. Currently, much of the fiscal and monetary policy is driven top down by the central bank and the government. A given policy choice need not be suitable for a whole country and at all points of time because of the inherent diversity of the constituents. By decentralising the policy conduct, mutualized IOUs can stabilise business cycles and improve economic wellbeing of all sections of society.

Novel asset tokenization to create new asset types

Conventionally, assets referred to real ones like properties and gold or financial ones like stocks, bonds, ETFs. Other objects of value like patents, warehouse receipts, brands are referred to as 'assets' in an accounting sense but are rarely traded or fractionalized.

Tokenization of such assets can enable wider access to them for investors. It can also enable novel forms of collaboration - for example a researcher can raise money for a project with a pre-sale of a patent token for the outcome of the project.

Enabling novel forms of collaboration

The limited liability company has remained the chosen form of business collaboration for centuries. While it has its benefits of standardization and regulatory reliability, it is somewhat constraining when it comes to collaborations in the information age.

While the theory of the firm in the 19th and 20th centuries was based on bringing together land, labor, capital and entrepreneurship, the firm of the 21st century needs a different set of ingredients. Most of the modern-day firms are based on bringing together human capital, machine intelligence and ecosystem. In trying to fit this in the conventional form, most entrepreneurs are forced to chase passive capital which they then use to pay employees and other ecosystem contributors to their business.

It is much easier to collaborate for individuals, machine-intelligence-owners and ecosystem participants directly through the tokens of ownership in the enterprise. This was brought out nicely in several coin offerings over the last few years. While crypto-projects are a good demonstration of this approach, it is by no means limited to crypto-domain. A non-crypto start-up, say in the consumer healthcare space, can equally well use this approach to build a business.

Way forward

Newrl is aimed at generalized non-crypto-native use-cases in mainstream finance. Hence it has some fundamentally new features in its architecture as a blockchain - including identity and legally robust tokenization. It also has other features that enable a more effective execution of the proposed objective of decentralized social finance - including trust network, templatized smart contracts and decentralized organization.

It is our hope and expectation that decentralized social finance will evolve into a broader decentralized economy that enables new ways of collaborating for individuals. This will expand the scope of our economic activities well beyond the straitjackets of firms employing individuals, borrowing from banks and raising money from institutional funds. A decentralized economy will be a much more exciting cornucopia of decentralized organizations with diverse objectives, participatory models and payoff structures for their participants, raising financial and human capital through a wide array of vehicles and participant-types.

The current blockchains have established the foundational technology of trustless transactions in a decentralized ledger. Newrl aims to be the next-generation blockchain that builds on these foundations to enhance the varying levels of trust that already exists amongst participants in a real economy, using smart contracts that make contract formation and enforcement frictionless and a trust network that deters against wilful misconduct while rewarding honest behaviour.

References

[1] Nakamoto, Satoshi, Bitcoin: A Peer-to-Peer Electronic Cash System, https://bitcoin.org/bitcoin.pdf, retrieved 14th January 2022

[2] Ethereum Foundation, Ethereum Whitepaper, https://ethereum.org/en/whitepaper/, retrieved 14th January 2022

[3] Savelyev, Alexander, Some risks of tokenization and blockchainizaition of private law, Computer Law & Security Review, Volume 34, Issue 4, August 2018, Pages 863-869

[4] Putnam, Robert, Social Capital: Measurement and Consequences, OECD, https://www.oecd.org/innovation/research/1825848.pdf, retrieved 31st December 2021

[5] Kaur, Gagandeep, Gandhi, Charu, Scalability in Blockchain: Challenges and Solutions, Handbook of Research on Blockchain Technology, 2020, Pages 373-406

[6] Eric, How much does it cost to deploy a smart contract on Ethereum?, https://medium.com/the-capital/how-much-does-it-cost-to-deploy-a-smart-contract-on-ethereum-11bcd64da1, retrieved 10th January 2022

[7] Reiff, Nathan, Why Centralized Cryptocurrency Mining Is a Growing Problem, https://www.investopedia.com/investing/why-centralized-crypto-mining-growing-problem/, retrieved 2nd January 2022

[8] OECD, The Tokenisation of Assets and Potential Implications for Financial Markets, https://www.oecd.org/finance/The-Tokenisation-of-Assets-and-Potential-Implications-for-Financial-Markets.pdf, retrieved 30th November 2021

[9] Harvey, Andrew, Your brain is a distributed system, https://medium.com/@mootpointer/your-brain-is-a-distributed-system-80aa1b6611c0, retrieved 10th January 2022

[10] Zhang, Shijie, Lee, Jong-Hyouk, Analysis of the main consensus protocols of blockchain, ScienceDirect, ICT Express 6 (2020) 93-97

[11] Dar, Ishaq Ahmad, Mishra, Mridula , Dimensional Impact of Social Capital on Financial Performance of SMEs, https://doi.org/10.1177/0971355719893499, February 24, 2020

Appendix 1: Proof of Trust Protocol Details

The PoT protocol uses a state-based approach. For validating any transaction, having the latest state is sufficient and each valid transaction updates the state.

This state should have following components, which can be maintained as tables in a database.

1. Blocks and transactions tables
2. Identity tables: wallets, personids
3. Trust scores by personid combinations (including network trust score where source personid corresponds to the network trust manager smart contract)
4. Tokens
5. Balances by wallet-token combinations
6. Contracts
7. Distributed organizations
8. Nodes and their liveness status
9. Stake ledger

The state can be changed by execution of transactions by each node locally. Transactions are of following types.

Type	Function	Signed by
1	Identity management – create new personid and wallet, create a linked wallet etc	Introducer
2	Token management – create a new token, issue more of an existing token, destroy token etc	Asset custodian / contract manager
3	Smart contract execution – set up a smart contract from template, deploy it, execute specific functions in it, destroy it (if applicable)	Various
4	Two-way token transfer – sender1 sends token1 to sender 2 and sender2 sends token2 to sender1	Sender1 and Sender2
5	One-way token transfer – sender1 senders token1 to sender 2	Sender1
6	Alter trust score – person1 alters the trust score of connection directed from person1 to person2	Person1
7	Update network liveness of a node	Node owner
8	Smart contract internal transaction	None

A transaction in Newrl has the standard format as below.

{“transaction”: “data”, “signatures”: [{sign1}, {sign2}]}

Validation of a transaction is for its signatures as well as economics e.g. for a transfer transaction, the sender needs to have adequate balance.

<u>Transaction handling</u>

In PoT, any node can broadcast a transaction to its peers. Receiving peers ignore transactions that they already have (checked using transaction id). For valid transactions, each node stores them in the local memory pool and transmits it onwards through a gossip protocol using the transport layer (by default, this is the libp2p protocol).

<u>Selection of block creation committee and block minting node</u>

PoT involves block creation by a chosen node and verified by a randomly selected committee. The minting block is selected randomly inside the committee, where the selection probability is proportional to its trust score. Additionally, each node is required to deposit a fixed number of valuable tokens of specified types (Newrl token or USDC stablecoins) in a dedicated smart contract that governs the protocol.

For arriving at consensus, the network undertakes the following steps in each round for a new block creation. The below process begins for a given block immediately upon the inclusion of the previous block.

We assume the following at the end of the state update after the latest block inclusion.

N = Number of total nodes in the network (from the nodes table)
e_i = Liveness score of the ith node (from the nodes table, currently set as 1 for all entries)
s_i = Network trust score of the ith node (derived from the trust score table)

$H_{committee}$ = Threshold of trust score for inclusion in the block creation committee (set at 0 at genesis)

H_{live} = Threshold of liveness score for inclusion in the block creation committee (set at 1 at genesis)

C = Size of the block creation committee (set at 10 at genesis)

The selection space S for block creation group is created is follows.

$$node_i \in S \; if \; s_i \geq H_{committee} \; and \; e_i \geq H_{live}$$

$$S = sort(S, ascending, peerid)$$

Since the state is same for all participants, S will be identical for them.

We select a total of C nodes in the committee from S.

$$S_1 = S$$

$$d_1 = random(seed = hash_{latest_{block}}) \times size(S_1)$$

$$node_j = node\ at\ d_j^{th} place\ in\ S_j$$

$$S_j = S_{j-1} - node_j$$

$$d_j = random(seed = d_{j-1}) \times size(S_j)$$

When *j* = *C*, the selection process halts. *C* nodes are now selected for the committee.

The block minting node is always selected from the committee itself through a random selection with the previous block hash as the seed. Since everyone is starting with the same state and using the same hash as the first seed, the choices of all committee members as well as the minting node for a block will be same for all.

<u>Block creation by the minting node and broadcast to committee</u>

The minting node is expected to create a block using the below-mentioned transactions.

1. Standard transactions
   The node minting a new block will incorporate as many transactions in its local memory pool as fit a single block - ordered in terms of their timestamp. It also transmits these transactions to the committee members, with the expectation that the committee members either already have these transactions or can update their local memory pool with them if they don't.
2. Block reward transaction
   The block reward transaction in each block creation automatically, with the block creating node as the recipient.
3. Network trust score transaction
   The minting node calls the network trust score manager smart contract with a selected set of receipts from its memory pool, which refer to previous blocks (not necessarily just the latest one). The smart contract consumes the receipts and updates network trust scores. Each block has exactly one transaction like this. It is further described in the 'repercussions of voting' section below.

The block reward and network trust score transactions are not separately signed but are deemed signed by the minting node through its signature on the overall block. This removes the need for validating these transactions by other nodes since these are not transmitted ahead of time anyway. The selected node also updates its local state. The transaction fees from all included transactions in the block are transferred to the treasury address of Newrl. The treasury uses its balance as a liquidity pool of the tokens allowed for the transaction fees.

The mining node broadcasts the block to the rest of the committee members with its own signature and receipt as follows.

```
{ block_data : {}
  signature : {}
  receipt : {} }
```

<u>Block format</u>

Each block in Newrl has the following format.

block_index	An integer
proof	Set at 0 for regular block and 42 for empty block
Status	Set at 1 for regular block, -1 for empty block created because invalid submission by block creator and -2 for empty block created because of time-out
creator_wallet	The actual creator of that block – sentinel node for empty blocks
expected_miner	Expected creator of the block at that index
committee	Committee of nodes that will signed off on its validity
timestamp	Time in seconds since the UNIX epoch
previous_hash	Hash of the previous block
transactions_hash	Root hash of the transactions
transactions_data	{standard transactions, network_trust_score_update_transaction}

“Transactions data” includes only the transaction id for standard transactions. The actual transaction data is not sent and instead is taken locally by each node using the transaction id in the received block. This avoids any malicious modifications in the transaction by the minting node and hence reduces the need to validate the transaction again by the receiving node.

<u>Block receiving and validation by other nodes in the committee</u>

When a committee member receives a block from the minting node, it first validates the block as shown in the box below.

This process leads to creation of a receipt, which looks as follows.

```
{“receipt_data”: {“block_index”:<index>,
                  “block_hash”: <hash>,
                  “vote”: 1, -1 or -2,
                  “address”:<address>},
 “signature”:<signature>}
```

```
if block_index != current_latest_block_index + 1:
    return (Status = False)
if previous_block_hash != hash(latest_block):
    return (Status = False)
if not valid(block_signature) or signer!= minting_node:
    return (Status = False)
synchronize(memory_pool, peers)
for transaction_id in transaction_ids:
    if transaction_id not in memory_pool:
        return (Status = False)
if not valid(network_trust_score_transaction):
    return (Status = False)
return (Status = True)
```

Normally, a vote of 1 reflects the True status as per earlier steps above and vote of -1 reflects False status. However, a node is at liberty to send a vote of -2 that indicates that it has insufficient information, or it does not want to vote. This is useful in avoiding the other committee members as well as the wider network waiting for a given member's vote if it wants to abstain. Allowing it to vote -2 reduces the confusion about whether a node is inactive or abstaining.

Each committee members transmits its receipt to all other committee members and also transmits received receipts to others, in case they haven't received those from the original sender for whatever reasons. It is expected that at the end of the block_creation_timeout, all live committee members will have a sufficient number of receipts to conclude on block validity.

Using receipts to validate a block

We assume the following variables.

p_i = f(s_i)

v_i = vote of i^{th} committee member

P = number of 1 votes

G = number of -1 votes

Z = number of -2 votes

The following steps are used to validate a block based on its receipts.

$$V^{+} = \prod_{i=1}^{P+G} prob_i$$

Where,

$$prob_i = p_i \ (if \ v_i = 1) \ and \ [1 - p_i](if \ v_i = -1)$$

$$V^- = \prod_{i=1}^{P+G} prob_i$$

Where,

$$prob_i = p_i \ (if \ v_i = -1) \ and \ [1 - p_i](if \ v_i = 1)$$

$$Y = C - (P + G + Z)$$

If $Y > 0$, there are pending receipts.

If $V^+ > V^-$, assume all the pending receipts are -1

If $V^+ < V^-$, assume all the pending receipts are 1

Calculate Q+ and Q- as follows.

$$Q^+ = \prod_{i=1}^{P+G+Y} prob_i$$

Where,

$$prob_i = p_i \ (if \ v_i = 1) \ and \ [1 - p_i](if \ v_i = -1)$$

$$Q^- = \prod_{i=1}^{P+G+Y} prob_i$$

Where,

$$prob_i = p_i \ (if \ v_i = -1) \ and \ [1 - p_i](if \ v_i = 1)$$

If $V^+ > V^-$,

If $Q^+ > V^-$, consider the block valid, else consider the outcome indeterminate.

If $V^+ < V^-$

If $Q^- > V^+$, consider the block invalid, else consider the outcome indeterminate.

The above algorithm ensures that even with partial receipts a node can make a decision about the validity of a block. The algorithm also ensures consistency of decision with arrival of new valid receipts. An “indeterminate” decision either changes to valid or invalid or remains indeterminate. A “valid” or “invalid” decision remains the same with new receipts as well.

Committee member action after confirming the validity/invalidity of a block post receipts

Each committee member confirms the status of the block in light of the receipts from other members, as described in the earlier section.

A. Indeterminate status:

   If the status is indeterminate for all live members (which is possible if only a minority of the committee members are active) after a time-out limit described below, the committee members mint an empty block (with timestamp of 1 second after that of the previous block) without any signature, create receipts of "1" locally and broadcast it within the committee. Then they move to next step of broadcasting the empty block (validated with an inadequate number of receipts.) In this case, they do not transmit the valid non-empty block created by the minting node.

B. Valid status:

   If the status is valid, the members broadcast the block data, minting node's signature for it and the receipts they have locally to their respective peers.

C. Invalid status:

   If the status is invalid, they broadcast the invalid block with the accompanying receipts. Then, similar to 1 above, they mint an empty block locally (with timestamp of 1 second after that of the previous block) with no signature and broadcast it amongst themselves to create receipts for it. This empty block with adequate receipts is then broadcast.

After this, the committee members update their state locally in case of valid block addition. For empty blocks, there are no state changes other than addition to the blocks table.

Time-outs inside the committee

It may happen that the node everyone considers chosen does not conclude so because it has a wrong or non-updated version of the state. It may also happen that the node chosen is simply inactive.

Also, while a valid block might have been minted, only an insufficient number of committee members may be active.

In the above instances, it is necessary for the committee to conclude that they cannot wait any longer and must move forward with minting of an empty block. This time period is a protocol level constant and is referred to as t_1. This is the time trigger for the indeterminacy referred to in the previous section.

New block verification by rest of the network

After committee members broadcast any block as described above to their respective peers and then onwards to the wider network, the block verification and inclusion/rejection by the rest of the network (i.e. other than the committee) is as follows. The first part is to validate the block using receipts alone. This is done as follows.

1. Validate receipts
    a. Must be from someone who is a committee member
    b. Must be signed correctly
2. Check for the block referred to in the receipt - check if the broadcasted block's hash matches that in the receipt
3. Try to validate the block using the received receipts as described earlier above.
4. If validity status is indeterminate, wait for more receipts.

The validation of block from first principles is same as that described earlier in the context of committee members.

Following are the next steps locally taken for each node upon receiving a block broadcast.

Type of broadcast	Action	Further broadcast?
Standalone valid receipts	Store in memory pool	Yes
Voted invalid block i.e. one with adequate "-1" receipts	Do not act on the block, store receipts and block in memory pool	Yes
Voted valid block i.e. one with adequate "1" receipts	Move to next step of block validity from first principles	Yes if valid
Voted empty block with adequate number of receipts.	Move to next step of block validity from first principles	Yes if valid
Empty block with inadequate number of receipts, within time-out period.	Wait for a time-out period to check if another block broadcast is received.	No
Empty block with signature of sentinel node, after time-out period.	If there has been no valid block with adequate number of receipts up to that point, move to next step of block validity from first principles.	Yes if valid and not ignored
Empty block with inadequate number of receipts, after time-out period.	Query a few randomly selected nodes from the network for a new block. If they send a block with adequate number of receipts, refer to above cases. If they send an empty one with sentinel node signature, refer to the above cases.	No

Non-empty block with correct signature but an inadequate number of receipts, within time-out period	If there is an empty block with adequate number of receipts, already received, ignore this non-empty block. Else, wait to receive remaining receipts. If adequate number of receipts is received, go to appropriate step in the above. Do not act on the block in either case, in terms of appending it to local chain and updating state, if adequate number of receipts not received.	No
Non-empty block with correct signature but an inadequate number of receipts, after the time-out period	Query a few randomly selected nodes for a new block. If a valid non-empty or empty block with adequate number of receipts or an empty block with sentinel node signature is received, go to the appropriate next step from above.	No
No block received at all even after time-out period.	Query a few randomly selected nodes for a new block. If a valid non-empty or empty block with adequate number of receipts or an empty block with sentinel node signature is received, go to the appropriate next step from above.	No

Time-outs in the wider network, outside the committee

There is a network-wide time-out window, represented by t_2 (> t_1 mentioned earlier), which is the outer limit of time for which a node is required to wait for receiving a valid block with adequate receipts from the block creation committee (even if it's empty).

After this time-out, to avoid the network getting stuck (e.g. in the instances of indeterminate empty blocks from a committee), specifically chosen sentinel nodes (typically with the highest trust scores) broadcast an empty block with their signature. If received after the network-wide time-out period and in absence of an adequate-receipt valid block until then, the recipients in the network broadcast this block onwards. This is unlike the indeterminate blocks which are not broadcast onwards as a rule.

This will not create confusion or forking because empty blocks are all the same, irrespective of whether committee sends them or a specific node. The purpose of this step is only to avoid network being stuck for want of a valid block (even if empty) to add. To avoid dishonest nodes from sending empty blocks at random, this action is restricted to a few chosen sentinel nodes which have very high trust scores. In any case, nodes can choose to locally add an empty block as well, after the time-out period. Hence there is no additional information in this action other than a confirmation that this block is meant to be empty. Also, empty blocks sent in addition to valid blocks with adequate signatures are ignored anyway and the sentinel nodes sending such blocks are removed from the sentinel list for subsequent rounds.

Post-validation process by a node

When a receiving node concludes that it is ready to process a received block, as per the previous section's method, it proceeds as follows.

1. It validates the block for its data from first principles, as described earlier. This is not based on receipts but on the actual transactions included in the block. This is same as the validation carried out inside the committee.
2. If the block is found to be invalid despite the receipts suggesting otherwise, the node rejects the block and does not add anything for the time being. It does not broadcast the received message forward either. Its expectation here is that the full network will arrive at consensus based on the validity of the block anyway and if the node itself has mistakenly considered the received block to be invalid, it can catch up with the rest of the network in subsequent rounds by querying its peers for the updated blocks.
   It may yet happen that the committee and minting node were dishonest, and they send an invalid block. In such cases, majority of the honest nodes will not update their local state and keep waiting. The sentinel nodes can send an empty block to get the network restarted.
3. If found valid, it appends the new block to its own chain. It also transmits the whole message onwards to its peers, as stated in the previous section.
4. After a successful update as above, a node also updates its local states according to the transactions in the new block.

In future, the block addition broadcast will also include changes in the state so that receiving nodes can also double check their state updates.

Repercussions of voting

Each process of validation of a block by the committee members results in the creation of a vote receipt by that member about the block being valid or not. This receipt has a reference to a block index and the hash of the block being voted on besides the actual vote and the signature of the voter.

These receipts are broadcast at first by their creator and then onwards like standard transactions by the others in the network i.e. each recipient confirms the validity of the signature and includes in the memory pool as well broadcasts onwards if it is valid. This way, most of the participants in the network have valid receipts of votes by committee members on a given block. While these receipts are used by the committee as well as the rest of the network for validating the current block, they are retained in the memory pool for subsequent processing in network trust score update transaction as follows.

At the time of new block addition, the minting node incorporates as many receipts as possible, with a preference to older receipts, inside the "network trust score update" transaction referred to earlier, calling the network trust score manager smart contract.

The call input reads as follows.

data = {$receipt_1$, $receipt_2$.... $receipt_K$}

The network score update algorithm uses the receipts with the following logic.

1. Validate the signature on the receipt again. If valid, check the block index referred to in it.
2. Refer to the archive of the right committee members for that block index.
3. If the receipt is not from any of the committee members, update the trust score of the sender of the receipt downward. This is an instance of someone attempting to pose as a committee member without being one.
4. If the receipt is from someone in the correct committee list, check the actual hash of the block index referred to in the receipt from the local chain copy. Follow a different process for an empty block and a non-empty block.
5. For a non-empty block, match this actual hash with the hash of the block referred to in the receipt.
6. If the hashes match, and the vote is "1", update the trust score of the sender of the receipt upwards as specified. If the vote is "-1", update the trust score downwards.
7. If the hashes do not match, and the vote is "1", update the trust score of the sender of the receipt downwards as specified. If the vote is "-1", update the trust score upwards.
8. In a special case of an empty block, the archives may have one discarded block with inadequate "1" votes and either adequate or inadequate "-1" votes. This block is evaluated for its validity. The receipts that voted for this block as "-1" lead to a trust score reduction of those nodes. The receipts that voted "1" for this block lead to a trust score increase of those nodes. This is the instance of indeterminate valid vote or minority being honest but majority in the committee forcing its way to an empty block.

This process is carried out by the rest of the network as well when that newly added block is considered valid by it. This way, the network uniformly updates the trust scores of all the voting committee members as well as any other nodes that may have tried to pose as committee members. It also updates for attempts at denial of service in the rare instances where the majority of the block creation committee happens to be dishonest, leading to addition of an empty block (as described in 8 above). The catch-up with the current block may vary as per number of receipts pending. However, the protocol is adjusted to incorporate more receipts during the periods of fewer other transactions. (This is analogous to how a human brain updates memories during sleep!)

Network Contribution Trust score

Each node on Newrl network is linked to a specific personId. The transactions and blocks broadcasted by the node are signed with one of the wallets mapped to this personId. The selection probability of each node for minting the next block is proportional to its network trust score.

Assume a network size of *N* nodes. We define a variable Δ = 0.2 by default. It can be reset by the Newrl community.

Every new joining node has a trust score of 10.

The trust score s_i of i^{th} node is modified by the trust score manager, using the receipts provided to it, as follows.

1. Each valid block creation by that node increases its trust score by a value as follows.

   $s_i = s_i + 10\Delta$

2. Each invalid block creation by that node decreases its trust score by a value as follows.

   $s_i = s_i - 100\Delta$

3. Honest receipts update the trust score as follows.

   $s_i = s_i + \Delta$

4. Dishonest receipts update the trust score as follows.

   $s_i = s_i - 10\Delta$

The value of s_i is capped at 100 and floored at -100. In state database, the value is stored with 4 decimals, so a score of 25 reads 250000.

The trust score change is asymmetric. For instance, it would take a node just 50 dishonest receipts to get from a score of 100 to 0 while it would take more than 500 honest receipts to climb back up from 0 to 100. In terms of blocks the same numbers are 5 dishonest blocks to go down from 100 to 0 and 50 honest blocks to make the journey back up.

Rehabilitating a node

It is possible that a node misbehaves and gets past the lower bound of 0 trust score to be included in any future committee. This would make it impossible for a node to be rehabilitated if it wanted to prove its honesty.

To deal with this situation, a rehabilitation smart contract can be called by at least *C* persons with trust scores in the top decile of the network and with a collective deposit of at least *K*C* times that of the standard node deposit (K set at 100 to start with). This transaction if picked up by a minting node will reset the trust score of the rehabilitating node at 1. The community can decide to use the deposit submitted here to compensate any loss from malicious behaviour of the rehabilitated node in future.

Byzantine Fault Tolerance and other security considerations in Newrl

Unlike the current public blockchains, the nodes participating in the network of Newrl are not anonymous. Secondly, their behaviour history is retained in the form of the network trust score. More specifically, a node creating a wrong block is immediately recognized before the block is broadcast, by the other members of the block creation committee. Since this committee is randomly chosen, the probability of malicious actors being able to reliably control its outcome is vanishingly small.

Assuming the proportion of dishonest nodes is *d*, even in absence of trust scores, the raw probability of a committee of C parties having a majority of dishonest nodes is given by the following.

$$p_{dishonest\_committee} = d^{(C/2)}$$

For d=0.33 and C=20, this probability is 0.002%.

Within a committee, since trust scores are used to arrive at validation of a block by its members, the trust_score informed votes have a vanishingly small probability of being manipulated by even a majority of dishonest nodes. If the honest nodes with high scores vote against a new block, the dishonest nodes' votes for the block can still be negated.

Even after a block is broadcast by a committee, it is open to audit by others in the network. Since others too additionally validate a broadcasted block and ignore it if found invalid, the attacker's work is not finished even after being able to add an incorrect block through majority control of a committee.

An attempt to vote maliciously against a valid block, with the objective of simply creating hurdles in the regular operations of the network (and prompting addition of an empty block) also gets caught fast enough as empty blocks prompt the network to review the discarded block and update trust scores - punishing majority if the block were valid and minority if it were indeed invalid.

The most reliable defence against malicious actors is that the community of Newrl participants is built organically with gradual addition of persons known to someone or other in the existing network. A new person cannot join the network without being invited by someone who can vouch for them. This promotes honesty without centralizing control of the network.

Lastly, since all wallets are KYC verified, an attacker even after succeeding to steal tokens will not be able to use the stolen tokens for anything without revealing their identity. It is akin to stealing money into one's typical KYC-ed bank account (as against the numbered bank account of the Swiss Banks variety, which are analogous to the current blockchains' addresses).

A Sybil attack is extremely difficult to employ on Newrl, for the following reasons.

a. Each node on Newrl has an associated personId and a KYC wallet. A sybil attacker will need to impersonate several persons along with their valid documents.
b. Each wallet on Newrl is added upon another person/institution vouching for that person. This reduces the speed with which multiple new accounts can be added by a malicious actor.
c. A new node does not have a very high probability of being selected for minting a new block. It has to patiently build good behaviour history over time.
d. Each new block added is still verified by others in the block addition committee which is chosen randomly for each block.

The denial-of-service attack in Newrl is avoided by throttling the number of transactions a node can receive from another node.

Overall, would-be attackers have to patiently build at least moderate trust scores through valid contributions over time and valid KYC at the start, to be even in the reckoning for being able to influence the outcome of any voting. Even then their attacks are thwarted by the multiple checkpoints on validity of transactions, blocks and receipts.

Appendix 2: Deriving the summary trust score of a participant

It would be useful to have a summary trust score based on the community's assessment of an individual. Such a summary trust score is derived as follows.

Start with community trust score estimate of 1.0 for all nodes.

$$C_i = 1.0$$

Carry out iterations (ITER times) as follows. ITER is a protocol constant and is set to 100 by default.

1. Start with the node that has the highest trust score in the current state. For equal scores, arrange nodes alphabetically in the order of their personids.
2. For the *i*th node with scores s_{ji} (score of node *i* in the assessment of node *j*), the community trust score is,

$$C_i = \frac{\sum_{j \neq i} C_j s_{ji}}{\sum_{j \neq i} C_j}$$

3. One iteration is finished when C_i for the last node is updated.

Calculate the scores with 4 decimals of accuracy. After ITER iterations, round the C_i values to two decimals and multiply the same by 100 to arrive at an integer value of the score.

Appendix 3: Micro-bank architecture in Newrl

Conventional banks have two major types of liabilities - equity and deposits, and two major types of assets - loans and liquid assets (cash and government bonds). A microbank follows a similar architecture. There are following participants in the microbank DAO.

1. Community depositors: These are individuals or small businesses that pool their money in the form of stablecoins to create the deposit pool for lending.
2. Borrowers: These are individuals or small businesses that borrow and create the loanbook.
3. Institutional debt investors: These are pools of capital from outside the community which evaluate the microbank's records in solvency and decide to contribute to the liability pool with their own money.

The balance sheet of a microbank on the liability side has deposits of depositors from within the community and senior debt of institutional debt investors from outside. The latter is senior in the sense of tranching of risk. All defaults in the loanbook first affect the deposits portion and only when it is exhausted, does it affect the senior debt. Given the lower risk, the returns on senior debt are lower than those on deposits.

The asset side of the balance sheet is primarily made of loanbook constructed as a pool of all loans of the microbank. There is also a small allocation to liquid assets like stablecoin or highly liquid and stable-in-price tokens, if any.

This is shown in the diagram below.

A microbank is characterised by specific rules on whom to lend and how much. These specifications are encoded in its lending smart contracts. These specifications are governed by the governance token of the microbank. Unlike conventional banks where equity holders have all the power, in a microbank, the governance tokens are spread out across depositors, institutional debt investors and borrowers. Hence all stakeholders get to vote on how a microbank's operations are run. To be clear, this is not execution of specific loans, which are anyway managed by predetermined smart contracts. This is about the overall framework within which the microbank operates. For example, how much liquid assets to keep aside, what is the upper limit per borrower, how long of a cure period a borrower should receive are some questions that the participants need to agree on and encode in the governance structure. This is open to revision based on votes of all the governance token owners from time to time.

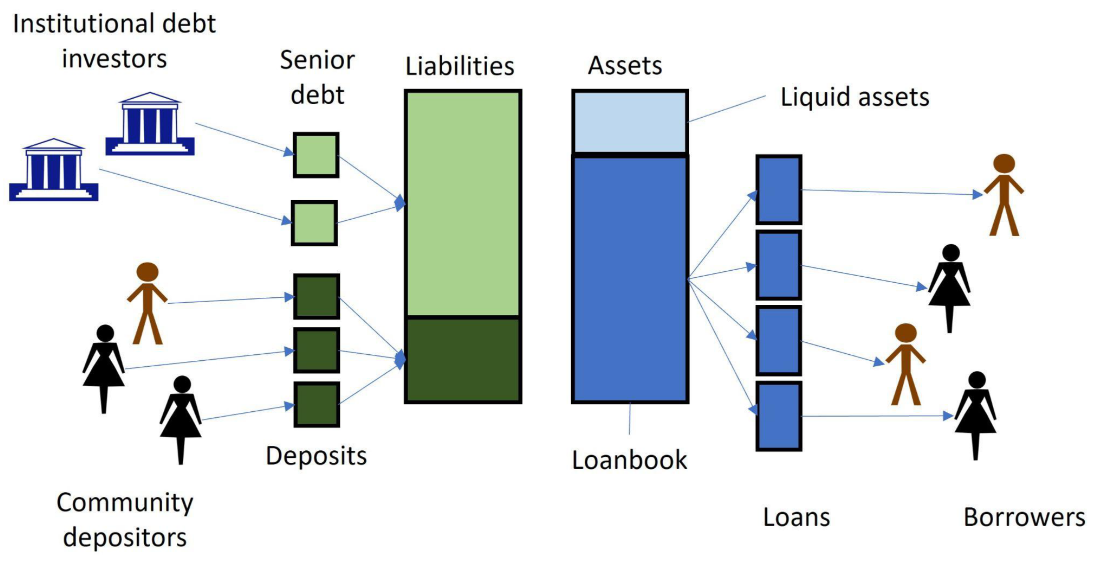

Diagram: Microbank architecture

The next phase of evolution of a microbank is an ability to issue its own money token. In the above description, the depositors and institutional debt investors brought stablecoins purchased/sourced elsewhere into the smart contract of the microbank DAO, which was then lent onwards to the borrowers. The DAO smart contract issues tokens of deposits to the depositors and tokens of senior debt to the institutional debt investors. These tokens are transferable. If the senior debt and deposits are traded at some price, presumably driven by the assessment of the broader ecosystem of the solvency of a given microbank, the bank can issue fresh liability tokens (either deposits or senior debt) directly in the accounts of the borrowers. They can then trade these liability tokens for stablecoins in the secondary market for these. That way, the microbank does not need to directly source from the depositors and institutional debt investors so long as they are happy to buy the microbank's liability tokens in the secondary market from the borrowers.

This comes close to the current operations of the banking system. The banks credit the account of a borrower with 'freshly issued credit money' which is then used by the borrower to pay for things. In the banking system, the price of all liability tokens of banks is constant at 1 unit of the fiat currency. This need not be a constraint in the microbanking system.

A further step in the evolution of microbanks would be the emergence of a 'central bank DAO' that consists of individual microbanks and the mutualization of their liability tokens. The central bank DAO can ensure smooth functioning of the system by creating liquidity for solvent microbanks' liability tokens as needed.

Appendix 4: Mutualization of tokens - details

The term “contributor” refers to persons owning assets who are participating in the process. The term “pool” refers to the collection of all tokens of all assets included in the process. Other terms have the conventional meanings associated with them.

Let us say there are N contributors to the mutualization pool. The ith contributor’s contribution to the pool in terms of number of tokens is Qi and the price of that token in a common currency (such as the US Dollar) is Pi.

The total value of the pool is

V = Sum(over all i, Pi*Qi)

If a total of Qmut tokens are created to represent the pool, the price of each mutualized token is,

Pmut = V / Qmut

The value that each contributor gets back is same as that contributed by her i.e. Pi*Qi, but stated in terms of mutualized tokens. If Ri represents the number of mutualized tokens that each contributor gets in return, it can be determined as follows.

Ri = Pi * Qi / Pmut

Simplifying further,

Ri = Qmut * Pi * Qi / sum(over all i, Pi * Qi)

A generalisation of the first-time mutualization is ongoing mutualization and demutualization. Consider the jth contributor looking to add Qj tokens of price Pj to an existing mutualization pool with mutualized token quantity Qmut and current value V. This value V is constantly updated for changing prices Pi in the expressions Sum(over all i, Pi*Qi).

The number of mutualized tokens that the incoming jth contributor receives is linked to the current value, current mutualized token supply and newly contributed value by her. The price of the mutualized tokens is maintained to be the same before and after this inclusion. This leads to the following result. Rj represents the number of mutualized tokens received by the jth contributor. It is determined as follows.

Qmut + Rj = (V + Qj*Pj) / Pmut

Therefore,

Rj = (V + Qj*Pj) / Pmut - Qmut

Next, consider the case of demutualization by an investor for the kth token i.e. someone returns a number Rgen of mutualized tokens and asks for a suitable number of the kth tokens in return. It is not necessary that this is the same person that contributed the original kth tokens.

The value returned is Rgen*Pmut where Pmut is the latest price of the mutualized token. Qk represents the number of kth tokens returned to this investor in return for her Rgen mutualized tokens and Pk the price of the kth token. After returning Qk of the kth token to the demutualizing contributor, the pool value is V - Qk*Pk and number of outstanding mutualized tokens is Qmut - Qk.
Since the price of each pool token needs to remain the same after the transfer, the below relationship holds.

$\text{Pmut} = (V - \text{Qk}*\text{Pk}) / (\text{Qmut} - \text{Rgen})$

$\text{Qk} = (V - \text{Pmut}*(\text{Qmut} - \text{Rgen})) / \text{Pk}$

The mutualization is carried out through a smart contract on the blockchain. This contract has the above mentioned logic embedded in it as a computer code. The contract can be called through blockchain transactions which carry as input the transfer to the contract of some number of tokens - either of a constituent or of the mutualized pool. The contract returns either the mutualized token (during a mutualization) or the specific constituent token (during demutualization). The contract can also include additional risk controls such as maximum and minimum proportion of each constituent in the total. It can be perpetual in time or have a definite maturity date upon which it automatically demutualizes and replaces mutual tokens with constituent tokens at the applicable exchange rates.
Like a typical smart contract on public blockchains, the mutualization smart contract is typically unalterable. However, participants can provide for alterability through consensus.