The Verified transaction protocol

Kallol Borah

Verified AG
kallol@verified.network

Abstract

The Verified protocol is a mechanism to transfer value among distributed applications, and for them to execute transactions between themselves. Such transactions are expected to be cryptographically secure, and will not be repudiated by other parties once transactions are committed. Commitment or confirmation of transactions should not also require a central broker between transacting parties. The technical implementation of the Verified transaction protocol uses an economic mechanism to administer a set of incentives on the Verified network. This involves the use of the Via digital currency whose description is outside the scope of this paper.

Keywords: blockchain, peer to peer, distributed systems, decentralization, trust

1. Introduction

In recent times, public blockchains such as Bitcoin and Ethereum, and private blockchain systems such as Hyperledger have tried to accomplish what Verified has set out to do. However, these experiments have been met with varying degrees of success. Public blockchains like Bitcoin work on broadcasted transactions and a hash power based confirmation of transactions bundled in blocks, which have led to significant transaction latencies, and enterprises not willing to participate in a network where transactions are visible to everyone. Proof of stake based blockchain protocols on the other hand introduce bias that get reinforced as the network rewards those peers that confirm blocks. On the other hand, permissioned distributed ledgers like Hyperledger use central transaction ordering elements which reduces the idea of a decentralized system where trust is built with consensus to a distributed database where trust is enforced with centralized components.

The design objectives of Verified are therefore to alleviate the issues of transactional latencies using a novel consensus mechanism that is also Byzantine fault tolerant, and to enable multiple transaction sub-nets to exist with the ability of these networks to merge at times when transactions between networks are required. The design of the Verified protocol also recognizes the different roles digital currencies and digital assets play and the different characteristics assets and currencies should have. This should allow tokenization of digital assets denominated using a stable digital currency such as the Via.

2. The Verified network organization

Each node on the Verified network has a 160 bit network address. The network address space therefore comprises 2^{160}-1 addresses expected to be collision resistant, and can be visualized as a binary tree with leaves representing nodes that are connected to one or more descendants that are prefixed with a common address. A user or peer on the network is identified by a digital certificate (eg, X.509) and is mobile across network nodes. In that sense, a peer identifier is a value contained in a node on the network. Multiple peers can get authenticated and log in to the same node. A peer can also be mobile across nodes and get authenticated and log in to multiple nodes, one after the other.

Look up for a peer by another peer is therefore a set of iterative hops. The number of hops is O(log $_n$), which effectively means a peer searches for the target peer by iteratively looking at the half of the network remaining after each look up. Each peer on the route to the target peer retains the address of the querying peer and acknowledges its presence. In this sense, the look up mechanism is symmetric and communication is asynchronous. Each peer on the Verified network maintains a bucket of peer addresses which is its routing table. Each route table entry is a tuple<IP address, UDP port, public key, nodeID=hash of key >. The number of entries in the bucket should be such that not all peer addresses in the bucket fail or become unreachable at the same time.

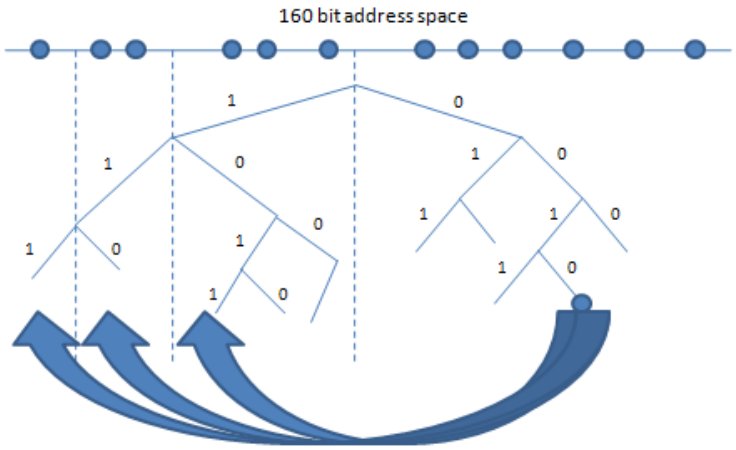

Figure 1. Binary network tree and look ups

The network organization builds up from the design of the Kademlia P2P protocol. The difference with Kademlia arises from the way Verified runs transactions on the network.

3. Transactions

Communications are full duplex and asynchronous between peers on the Verified network. To communicate and transact with a peer, its public key and network address should be available to the peer that seeks to transact with it. The transaction initiating peer encrypts its message to the target network peer using the target's public key. Once it receives the message, the target decrypts the message using its private key. The target then acknowledges the receipt and responds by encrypting its message to the initiator using the initiator's public key.

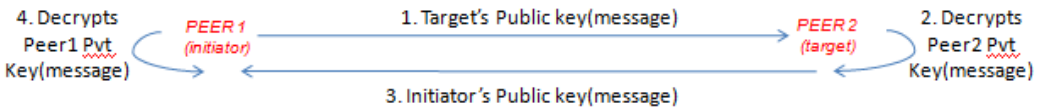

Figure 2. Full duplex, encrypted communications

It is reasonable to expect that the implementation will take care of NAT traversal, communication using secure network protocols, and network fault tolerance such that any communication failing to reach the target will be retried till a TTL expiry is reached.

However, communication between peers by themselves does not constitute transactions on the Verified network. Transactions are hashes of request/response pairs that both peers to that transaction have at the end of the duplex communication cycle between them.

Transactions need to be stored on the network by the initiating and target peers, but be first certified by a network node that is reachable by both initiating and target peers. So, it is not necessary that a peer is logged into a node for it to qualify as a certifying node. To select the certifying node, both peers look at their buckets and select the oldest network node in their buckets. Once this is done, both network peers look up the certifying node and send it their transaction hashes. The certifying node matches the transaction hashes it receives from the initiating and target peers and if they match and transaction is confirmed, the certifying node iteratively uses the same hops that the initiating and target peers used to reach the certifying node to record the confirmed transaction with its timestamp.

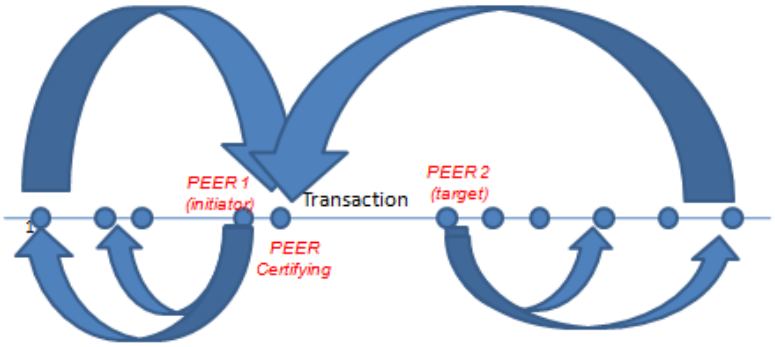

Figure 3. Routing to the transaction certifying node

The certifying node hashes the matched transaction with the previous certified transactions linked to other certifying nodes. This creates a chain of ordered timestamped transactions on the network that are all confirmed. The certifying node, the initiating and target network peers, and the network nodes on their route to the certifying node all store the confirmed transaction hashes.

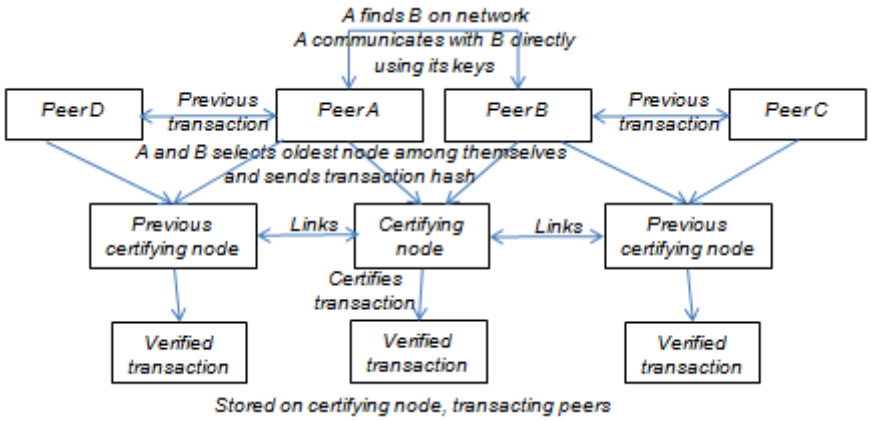

Figure 4. Chain of certified transactions on the Verified network

It is worthwhile noting that there could therefore be multiple transaction chains on the network at any point in time. Each transaction chain would involve a common peer and other peers that it transacts with. In the illustration above, the certifying node for the transaction between peers A and B becomes a part of two transaction chains – one with previous transactions for A and another with previous transactions for B.

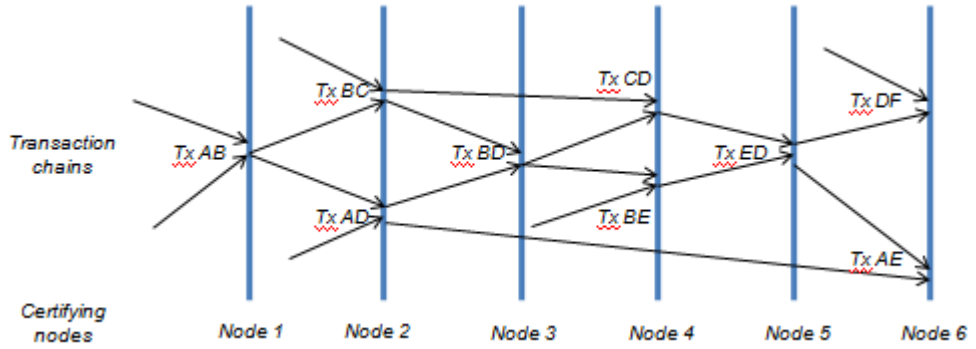

Fig 5. Interleaving transaction chains across certifying network nodes

As a transaction chain grows, it will also mean that the transaction chain with each addition of a new link to it may be replicated across a different set of nodes connected to different certifying nodes. Updates to the transaction chain need not be replicated to nodes containing older versions of the transaction chain.

Any transaction in the chain can be verified by checking if its certifying network node contains the transaction and if peers connected to the certifying node contains the transaction. Any attempt to tamper with even one transaction in the chain will render succeeding transactions executed by the peer incapable of being verified successfully.

It also needs to be noted that the number of replicas of any transaction can be on varying number of network nodes which serves as a security feature in the sense that an attacker cannot be sure it has removed all references to a transaction completely from the network.

Using the Kademlia protocol also means that the location of nodes do not determine their ‘closeness’ on the network and serves as a deterrent for forming groups of attackers.

4. Transaction Storage

Each peer on the Verified network stores certified transactions it is part of in a Merkle tree. Even if the peer logs in to multiple network nodes on the Verified network, it downloads and synchronizes its transaction store from previous nodes. A certain number of transactions are bundled together in a block by including the hash of the merkle tree root of which they are a part. Then, instead of storing all transaction hashes on the node, storing the hash of the merkle root or the block’s header is sufficient to track transactions on that node.

Each transaction on the Verified network is a tuple <transaction identifier, current transaction hash, transaction type, block identifier, certifying node identifier, participating peer identifiers, timestamp>, where the transaction identifier is the hash of the current transaction seeded with the previous transaction in the block. Transaction types could be SEND, RECEIVE, SWAP, MULTISIG, etc. The transaction itself is nothing but a hash of the name/value pair transferred from the sender to the receiver. Example of a name could be an asset name, and the value being the asset identifier. In the case of a currency, the name could be the currency’s name, and the value could be the amount transferred.

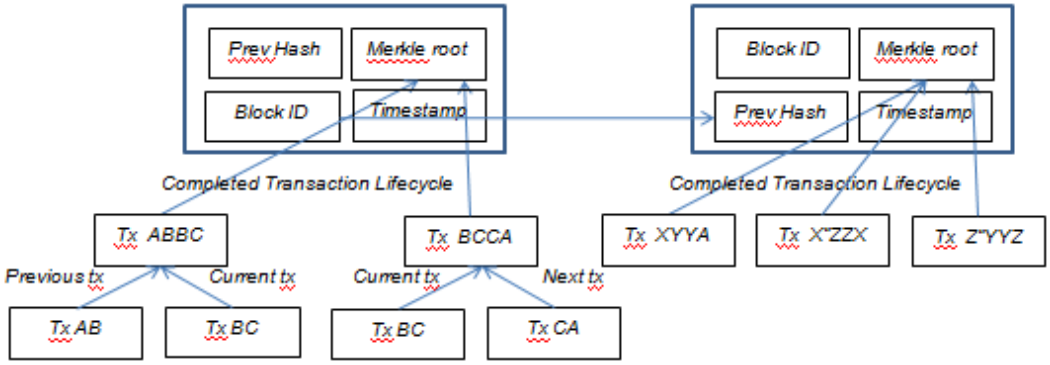

Figure 6. Merkle tree based transaction storage

It is important to note that blocks are chained on each network peer but they are not chained across peers as is common in other blockchains. Instead, it is transactions that are chained on the network. Blocks are merely a unit of storing transactions efficiently. Chaining the blocks also ensure that if a network peer tampers with a transaction or a block, succeeding transactions in following blocks won't be capable of being verified successfully even if such transactions of the peer are with different network peers.

5. Consensus

For network peers to reach a consensus on a transaction, they must agree that the transaction is what was executed between them in the first place. A consensus mechanism must therefore have the ability to distinguish a transaction where the parties to the transaction give a different account of what transpired between them.

The second leg of a consensus mechanism has to establish that none of the peers to the transaction committed the same transaction defined as a transfer of the same asset with the same timestamp with another peer previously. This is commonly referred to as 'double spending' in related literature.

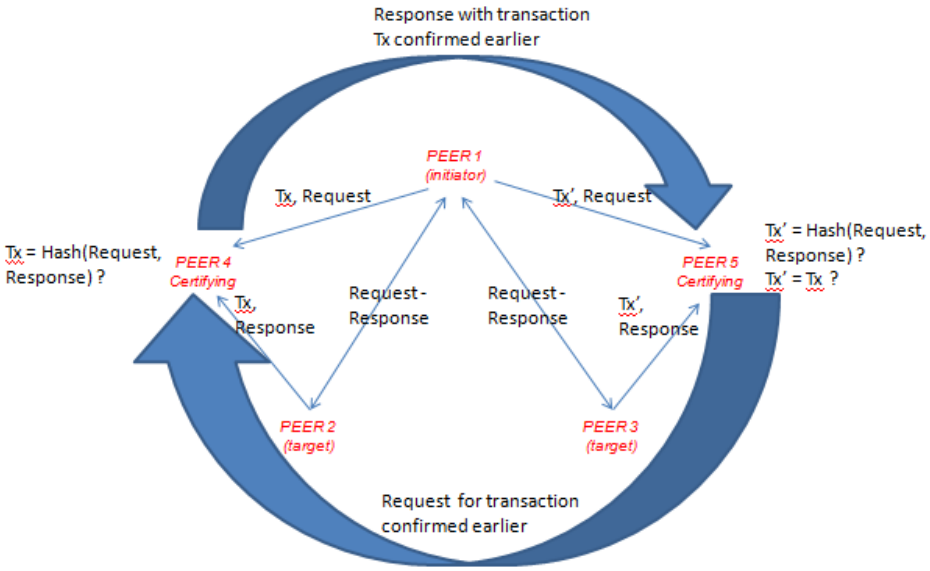

Figure 7. Coordination between certifying network nodes

On the Verified network, the first leg of consensus is established with both peers to a transaction sending the transaction which is a hash of the request and response, and each

peer sending its own message which could be the request or the response to the other peer with which it is transacting. The certifying node compares the transaction hashes received from both peers, and then checks if the transaction hash equals to the hash of the request and response received separately from each of the peers. If either the transactions hashes sent by the network peers are not the same, or if the transaction hash created with the request and response sent is not the same as the transaction hashes sent by either peer, then the transaction is not certified and the certifying node does not include it on the Verified network by hashing it and its timestamp with the hash of the previous certified transaction.

The second leg of consensus on the Verified network involves the certifying node requesting the network peers to provide it with certifying peer addresses for previous transactions they have executed with each other or other peers. The certifying node to the current transaction then requests nodes at those addresses for the previous transaction details or simply the hash of the transactions they hold with them. The certifying node then checks if the current transaction is contained in any of the previous transactions and if that is the case, the current transaction is not certified and vice versa. Otherwise, the certifying node confirms the transaction, keeps a reference to the previous certifying node, and passes its own reference to the previous certifying node to store. This forms the transaction chain with each link in the chain containing - chain element identifier (could be hash of previous element in the chain with this one), transaction hash(id, value), sender, receiver, timestamp, previous certifying peer, next certifying peer.

If certifying nodes on the Verified network go out of the network, transactions on the network can still be certified using copies of the transaction chain left on network nodes on route to the certifying nodes.

6. Ordering of transactions

While certifying transactions using the two legged consensus mechanism on the Verified network addresses the basic conditions for Byzantine fault tolerance, the system needs to consider the situation when one or both peers to a transaction won't report their previous certifying node correctly. This is where it is important to understand how transactions are ordered on the network and how this ensures tolerance against dishonest behavior.

The first certifying node in the network is always the root node on the network. In case, any peer does not report any certifying node, the root node is by default considered as the certifying node. If the root node does not contain a transaction with the peer, then the peer's subsequent transaction is not certified.

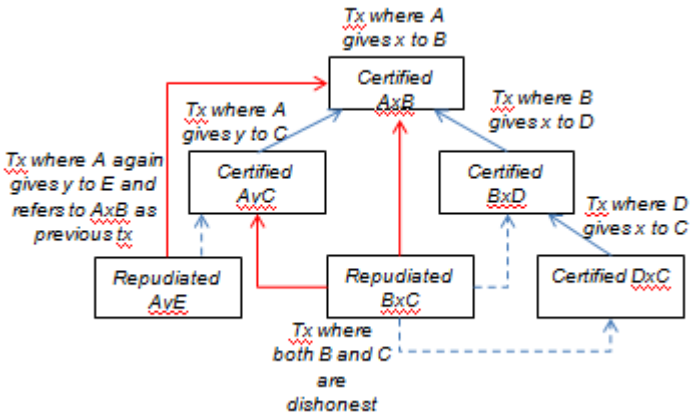

Figure 8. Dishonest reporting of previous certifying nodes

The above illustration shows transactions where one or both network peers are dishonest about reporting previous certifying nodes.

The first case is where peer A in its transaction with peer E does not report its previous certified transaction where it gives y to peer C, which peer A again subsequently gives to peer E. In this case, the certified transaction AxB provided by the dishonest peer A leads on to the next transaction AyC where peer A gives y to peer C, which conflicts with the transaction AyE which is then not certified.

The second case is where peer B gives x to peer D which in turn gives x to peer C. Then, in the transaction BxC, both peer B and peer C turn out to be dishonest. While peer B refers to the transaction AxB as its previous transaction and does not reveal that it gave x to peer D before this transaction, peer C refers to the transaction AyC as its previous transaction and does not reveal that it had got x from peer D before this transaction. In this case again, both certified transactions AyC and AxB can trace to transactions BxD and BxC which repudiates the transaction BxC which is then not certified.

Such ordering of certified transactions also prevent forks from occurring on the Verified network where one or more peers go about extending a chain of transactions that emanate from dishonest or wrong reporting of previous transactions.

7. Verifying transactions

To verify transactions on the Verified network, a network peer should identify the merkle tree branch linking the transaction to the block it is packaged in on a participating peer.

In order to ensure that participating peers include all transactions in a block or not succeed in making certifying nodes skip transactions included in a block in a transaction chain, the Verified protocol requires certifying nodes to query the participating peers for their recent transaction block. From the most recent transaction element in the block, the certifying node can check the previous certifying node and query the previous node for its transaction chain. The previous certifying node's copy of the transaction chain should be the same as the most recent transaction block the certifying node retrieves from the peer. If the previous certifying node is not the most recent one, it will also be pointing to a different certifying node succeeding it.

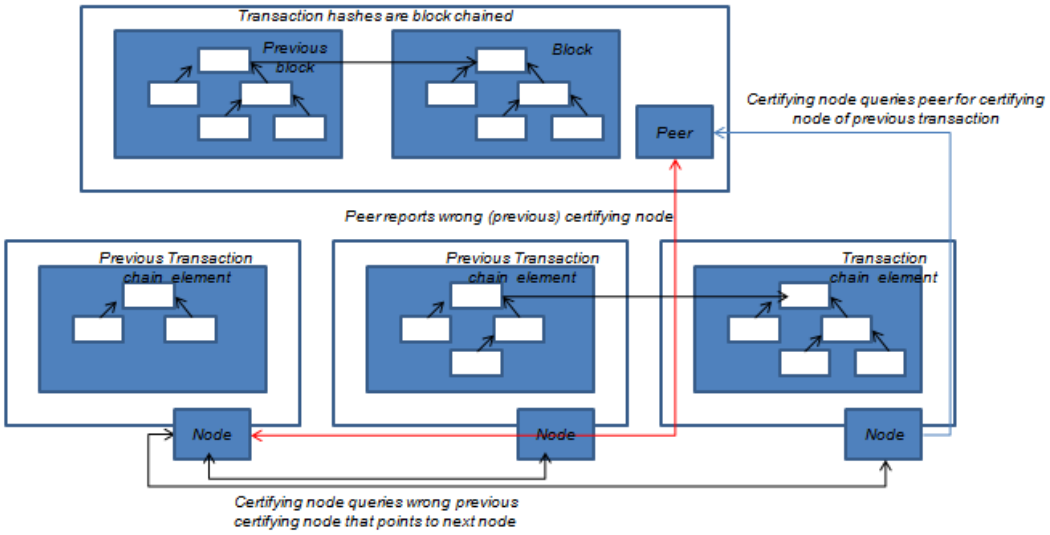

Figure 9. Verifying transaction chain before certifying transaction

If the previous certifying node is corrupt or is hacked by the peer in any way, querying a few previous certifying nodes should establish if the peer is honest or not.

To check circular transactions where an asset or commodity is transacted between more than two parties in such a way that the asset or commodity comes back to the parties again, a certifying node can check for the transaction in the transaction chain. The same transaction type with the same transmitting peer and the same timestamp and the same name/value pair for the asset should not be found in the transaction chain.

If the transaction involves expenditure of currency, the sum of currency received in all previous transactions should be more than the sum of currency expended in all previous transactions, and the difference between them should be less than the amount of currency proposed to be incurred as expenditure in the current transaction.

8. Privacy

By certifying users on the Verified network, Sybil attacks are expected to be discouraged where one or more users clone themselves and attacks the network. Verified network peers can have mappings to X.509 certificates.

Since routing hops on the network are iterative rather than recursive, it also prevents network peers in a route from knowing each other. This keeps transaction references distributed in the network address space and prevent related peers in a transaction or on the same route from wiping out all references of a transaction.

Since peers on the Verified network do not have to do broadcasts to announce transactions, chances of corrupting buckets or routing tables using network broadcasts is minimized.

Finally, besides using TLS like network security protocols, one way SHA hashes, etc, request and response messages exchanged in clear format should be homomorphically encrypted so that no message parts are in clear format in network, storage or volatile memory.

9. Inputs and Outputs

Transactions on the Verified network are expected to transfer value from one network peer to another. Therefore, each network node on the Verified network should implement a value store which can store assets such as investment certificates, credit notes, deposit certificates, etc, and currencies such as the Via, Bitcoin, US dollar, etc. Assets can be brought or acquired against currencies. Assets can be redeemed against currencies that are then received in the node's value store. Users on the Verified network can request Via to be transferred to a network address against Assets they hold on their network nodes.

A typical request format for transferring Via would be

transfer(type, asset reference, address to credit, amount of via)

Where, type is the type of transfer which can be a debit or credit transfer, asset reference is the certificate in the store of the peer, address to credit is the network address of the beneficiary peer, and amount of Via refers to the value of Via to transfer.

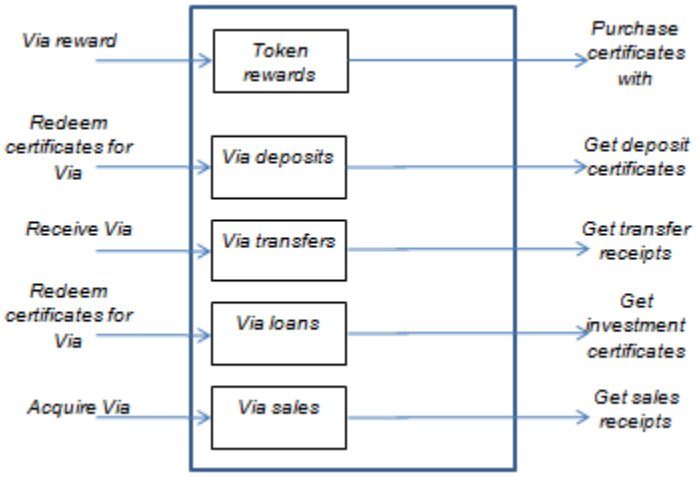

Figure 10. Value store on Verified network peer

APIs provided by each network peer on the Verified network should support the following functions : PING, STORE, FIND_TX, FIND_PEER.

Assets can have any number of sub types - eg, investments, deposits, loans, credit linked notes, etc. Sub types can be defined by developers/users. Each of these sub types can define their attributes such as - face value, coupon, issue date, expiry date, etc.

The functions on an asset could be
1. issue(asset_type, currency_amount) returns asset_id : this enables peers to create an asset in the asset store in exchange of a currency in the currency store.
2. redeem(asset_id) returns currency_amount : this is simply the mirror of the issue function. It returns back the currency to the currency store and removes the asset.
3. valueOf(asset_id) returns float value : returns the value of the asset in currency terms.
4. send(asset_id, to_peer_address) returns tx_id : sends the asset from one peer to another peer. Debits one, and credits the other.
5. transfer(asset_id, to_peer_address) returns tx_id : sends currency equivalent of asset to destination peer.

Currencies can have any number of sub types also - eg, via (which is our currency), bitcoin, ether, rupees, dollar, etc. Transfer types of a currency can be - credit or debit - type.

The functions on the Via currency type are
1. send(currency_amount, to_peer_address) returns tx_id : sends currency from one peer to another peer. Debits one, and credits the other.
2. deposit(currency_amount, term_in_days) returns acknowledgement_id : locks currency in currency store for specific number of days.
3. lend(currency_amount, term_in_days, coupon_rate) returns acknowledgment_id : debits one peer, credits the other for a specific term.
4. redeem(acknowledgment_id) returns currency_amount : returns currency against deposit or loan.
5. swap(first_currency, second_currency, first_currency_amount, second_currency_amount) returns tx_id : enables swapping of currencies.

It needs to be noted that the functions returning tx_id or acknowledgment_id or asset_id (collectively referred to as reference_id) simply locks an account for a quantum of

asset/currency, and releases and unlock/credits the appropriate account only when the certificate (ie, certified transaction) returned by the certifying node is used to call release(reference_id, certificate).

10. Incentives

Unlike other decentralized transaction protocols where work effort expended or stake committed enables peers to confirm transactions, nodes on the Verified network can only confirm transactions if they have been around on the Verified network for so long that they are selected by transacting peers to be certifying nodes. Certifying nodes are thus rewarded with Via equivalent to a small percentage of value they help transfer. These transfers could be debit transfers such as payments, or credit transfers such as loans to other network peers.

For availing incentives and rewards for certifying transactions or economic activities such as deposits, transfers and loans on the Verified network, each network peer has to consult an oracle that administers such incentives for certifications.

Private functions that need to be implemented on the Verified network for each network peer are

1) valueOf – purchase/sale, issue/redemption, lend/borrow and send/receive functions for assets and currencies on every network node should invoke this on the oracle hosted at the Verified network's root node. The value of the transaction, currency or asset type, and transaction type should be parameters that are inputs to this function.
2) setValue – this function should be implemented on network nodes on the Verified network that could eventually certify transactions. This function will be invoked by the Verified network's root node to set the value of Via issued as incentives to the certifying node by the Verified network's oracle.

Appendix

Formal proof of Verified consensus mechanism - TBD.

Acknowledgments

I thank Navin Sinha, my friend and collaborator on implementing the Verified transaction protocol for his insightful review and the list of questions that it produced.

References

[1] Satoshi Nakamoto, "Bitcoin: A peer to peer Electronic cash system", **(2008)**.

[2] Petar Maymounkov, David Mazieres, "Kademlia : A peer to peer Information system based on the XOR metric"

[3] Leslie Lamport, Robert Shostak, Marshall Pease, "The Byzantine Generals Problem", SRI International.

