**Verified : a new consensus protocol for decentralized applications**

**By Kallol Borah**

A lot of applications we use today are distributed applications where users are on the network and they communicate with a central server or groups of servers. Examples include search engines, social media sites, banking applications, and many more. Central servers means that there is a central authority that control access to other users on the network, have access to user data and is a single point of failure on the network.

To get around these drawbacks, applications can be designed to be completely decentralized. In such a scenario, there would be no centralized authority and instead, users on the network will coordinate with each other to carry out the functionality of the application. Examples include trading of financial products without a centralized exchange, sensor based applications to coordinate self driving vehicles, and so many more. In such applications, the challenge is to ensure that dishonest users are detected and their transactions to hijack the system nullified, and to also ensure that transactions are ordered properly so that resources are allocated to honest users appropriately.

A new class of applications based on blockchain technology have implemented multiple mechanisms to address these decentralized system requirements. However, most blockchain based systems rely on users broadcasting their transactions to all other users on the network so that they can form consensus on admitting transactions and such broadcasts slow down the system as the network of users grow. Another common feature of a lot of blockchain systems is that they adopt a technique called Proof of work which requires users who are also called miners to solve a complex problem before confirming transactions submitted by other users on the network. Such mining operations make blockchain systems expensive to operate since they consume energy and also slow down the process of confirming transactions.

We want to resolve these two challenges – first, of scaling the network without resorting to global broadcasts, and secondly, of reducing the cost and time taken for transactions to be confirmed – in a decentralized network where we presume there will be dishonest and bad actors.

The Verified consensus protocol is a new mechanism of coordination between users on a decentralized network such that transactions can be confirmed without having a centralized authority and without broadcasting them or requiring users to undertake expensive operations such as mining. The Verified consensus protocol applies to a network or system with users running decentralized applications on devices such as mobile phones and PCs, but can also apply to a network or system where devices such as sensors, controllers and actuators in automotive, home automation, industrial automation and other machine to machine applications coordinate among themselves without human intervention.

Users or devices on the Verified network would be on a network of some topology. This topology could be a star or ring based network. There would be some mechanism to discover participating users and devices, referred to as peers on the network by other peers. Discovery mechanisms could include broadcast based discovery or even use mechanisms such as distributed hash tables to optimize the number of hops one peer would require to find a target peer. Once a target peer is discovered on the network, the initiating peer may use its private digital key to sign its message to the target peer and then encrypt the signed message with the public digital key of the target peer. Once the message reaches the target peer, it can decrypt the message using its private digital key and check with the initiating peer’s public key if the message was signed by it. A peer also stores the addresses of target peers it finds and other peers on the routes to such target peers in a routing table. The routing table at each peer comprises random other peers which keeps getting updated.

The Verified consensus mechanism operates in two parts.

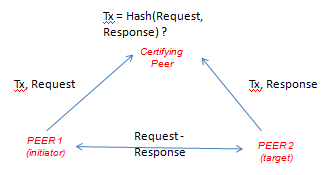
The first leg of arriving at consensus for a transaction requires the participating peers (users or devices) to first find and agree on a third peer (also called the certifying peer) that is the oldest (that is, it has been existing for the longest time) peer the participating peers know among themselves (that is, the certifying peer is either in the routing table of one or both of the participating peers). The requestor (peer that sends the request) then sends the request and the hash of the request and response received by it to the certifying peer. The recipient (peer that receives the request) sends the response sent and the hash of the request and response (it sends back to the requestor) to the certifying peer. The certifying peer first compares the hashes received from the requestor and recipient, and then it makes a hash of the request and response separately received from the requestor and recipient and compares the resulting hash to each hash received from the requestor and recipient. If the hashes match, the certifying peer can assert that the participating peers (requestor and recipient) agree on the request and response each sent to the other. This leg of the consensus process is referred to as certification, as the transaction is certified as being one that has taken place and is agreed upon by the participating peers. Refer to figure 1 for the certification workflow.

However, certification does not itself ensure that the participating peers are not able to leave out or misreport transactions to subsequent transacting peers. The state of any peer on the network which could be its account balance or state of other variables it holds is a result of the order of transactions it carries out with other peers. Thus, if the order or the number of transactions are altered or transactions that have occurred are taken out and not reported, the state of a peer cannot be verified. This therefore requires a second leg of verifying a transaction, that is, to make sure that a transaction is capable of being checked for authencity and leads to the current state of the peer that has executed it.

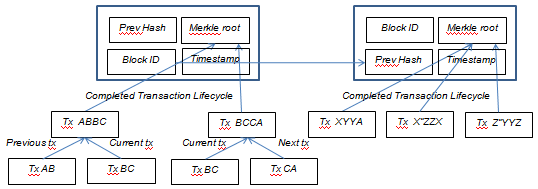
The participating peers store verified transactions in a data structure called the Merkle tree (figure 2). The Merkle tree can be visualized as a binary tree with leaves representing transactions that are connected to one or more descendants that are prefixed with a common part of their transaction identifiers. Each transaction on the Verified network is a tuple <transaction identifier, current transaction hash, transaction type, block identifier, certifying peer identifier, participating peer identifiers, timestamp>, where the transaction identifier is the hash of the current transaction seeded with the previous transaction in the store. The certifying peers also called verifying peers store verified transactions for each participating peer in a separate Merkle tree. The merkle tree on each verifying peer is linked with the transactions on previous and subsequent verifying peers.

The second leg of verifying a transaction requires each of the participating peers to send the hash of the root of the Merkle tree that stores its transactions to the verifying peer along with the data required to certify the transaction. The verifying peer downloads the most recent transactions from the tree and gets the references to the previous verifying peers for the most recent previous transactions. The verifying peer then requests each of the previous verifying peers to send the hash of root of the Merkle tree where it has stored the participating peer’s transactions. The hash of the Merkle tree root sent by the most recent previous verifying peer should match the current hash of the root of the Merkle tree sent by the participating peer. Similarly, hashes sent by other previous verifying peers should match the hash of the merkle tree root sent by the participating peer if the subsequent transactions are stripped away from it. This process ensures that transactions are ordered properly and any transaction can be verified by checking the hash of the Merkle tree of verified transactions. Leaving out previous transactions is also caught out since each verifying peer is linked to the next one. Collusion between participating peers and verifying peers does not take place since verifying peers which are the oldest existing peer between transacting peers keep changing as routing tables for transacting peers keep getting updated from time to time. Refer to figure 3.

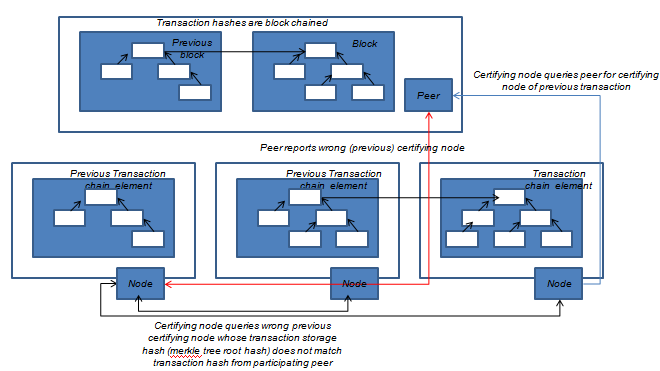
In case of transfers where the correctness of value transferred also needs to be checked, the verifying peer checks the transaction type for the transaction and verifies if previous transactions of the opposing type supports the current value transfer. For example, if the type is ‘SEND’, the verifying peer checks if the SEND value is more than the values of previous RECEIVE transactions in the Merkle tree based transaction store. Once the transaction is verified, the verifying peer sends the transaction identifier and timestamp of verification to the participating peers for them to store.



**Figure 1. Certifying a transaction : first leg of Verified consensus**



**Figure 2. Merkle tree based transaction storage**



**Figure 3. Verifying transaction chain by verifying peer**