Storage-centric mutable Blockchain in a network of trusted nodes

Abhijith Rajeev (Abe)

Problem statement: There is a huge network of nodes, all the nodes are in a mesh network topology sending data to the centralized data server, the whole application architecture needs to be redesigned to make it a decentralized data storage network to make the data more persistent and the application more reliable. The real time application has several nodes, which sometimes move to a remote location to gather data. There is a higher possibility that the remote locations don't have strong internet services, even in this case the remote nodes need to update the data so that all other nodes on the network can have access to it.

Blockchain based approach is a perfect solution as the nodes need a consistent network availability and even the data collected by those nodes at no-internet locations need to be updated onto the data storage blockchain.

Note: The mesh network architecture helps the node to update the data even in the event of network / internet unavailability. All the nodes are in a mesh network are sharing the data among themselves and to the centralized database server.

The appropriate solution here would be to achieve the off-line syncing of data among devices/nodes.

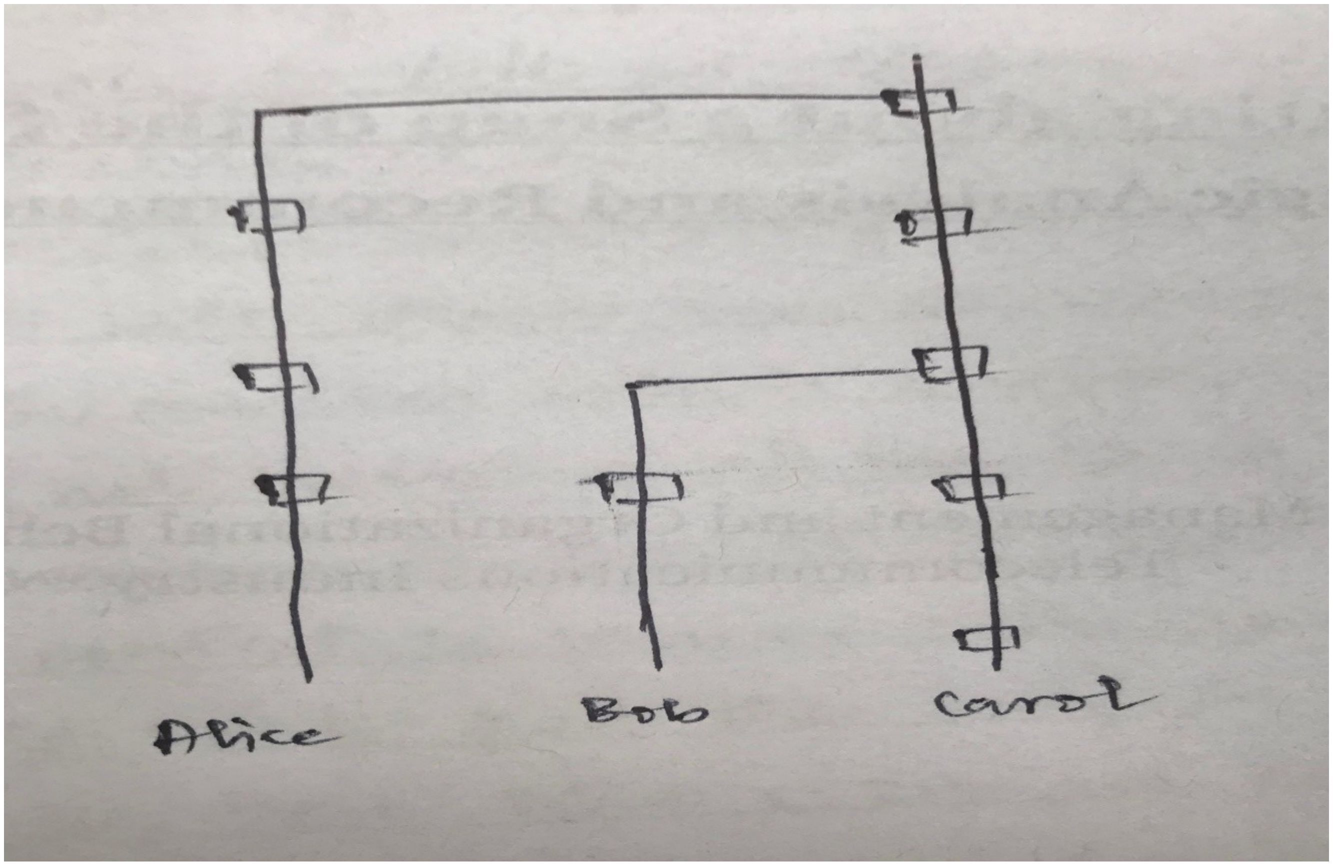

Figure 1

- In the above image, the small boxes can represent the blocks. Alice, Bob and Carol are the nodes which has these blocks. After one point of time, the blocks framed at the end of Alice and Bob are different as they have a branched out from the main blockchain and regular sync or the plain method of off-line sync can't be used here as it wouldn't result in full sync of the blockchain data. The methods to sync will be discussed in this document later.

Note: Since the chains at Alice and Bob are different from the main network they can be easily confused as sidechains, but that wouldn't be the conventional **sidechain** of blockchain. They are basically nodes which are lagging with respect to the block height. But they are not the conventional blockchain nodes as they have a different set of information compared to the main net. Syncing them to the main net needs a special approach than regular sync.

- **Offline Sync:** All the blocks in the chain are given the number and the node with the highest blockchain number / height will have the power or authority to start framing the next block by publishing the transactions. Here the node Carol have the highest block height, it has the power to start publishing new transactions to the main net. But the nodes Alice and Bob need to be able to add blocks to their chains even though they are not completely in sync.

- One approach would be to create **reserved blocks** in the main net. Since we know that the node with the highest block number acts as a miner and have authority to either publish or not to the main net. This idea can be implemented with little twist; the owner ship of the block can be delegated to a node, let's say Alice and only Alice can update / publish data to that block.

- Whenever nodes such as Alice and Bob are moving to the locations where they don't have proper connectivity to the main network, they can request the ownership of a block or two and a **blankspace** with the block head and block root will be created, and that block can only be filled or completed by the owner node.

- The owner node of these reserved blocks can push the blocks to main net when they come online, and the owner nodes need to be restricted in such a way that they shouldn't be able to push any data to the main network until the reserved blocks are framed and added to the main chain.

- This even helps in the situation when there is no internet connection, the nodes Alice and Bob can sync via Bluetooth or any other means when they come closer. The node with the highest block height and the node with the reserved block can exchange data among each other, creating a local consensus. And when they get the complete connection, they can take actions and sync with the main network / chain.

Note: The above-mentioned approach is graphically described in figure 2.

With reference to above mentioned approach and to figure 2.

- This approach would require some modifications to the traditional blockchain approaches, as the reserved block size need to be kept variable until that block is added to the main chain. And the block head / root hash need to be generated with no data in the block, the conventional approach of Merkle trees inside the block need to be modified.

- Instead of Merkle tree, a plain DHT (Distributed hash tree) approach can be used as the block content, which has the address hashes of the data stored on the cloud or a distributed storage unit. Proof of elapsed time; popular algorithms like Lamport timestamp can be used to keep track of the event time and to generate the root hash of the reserved block.

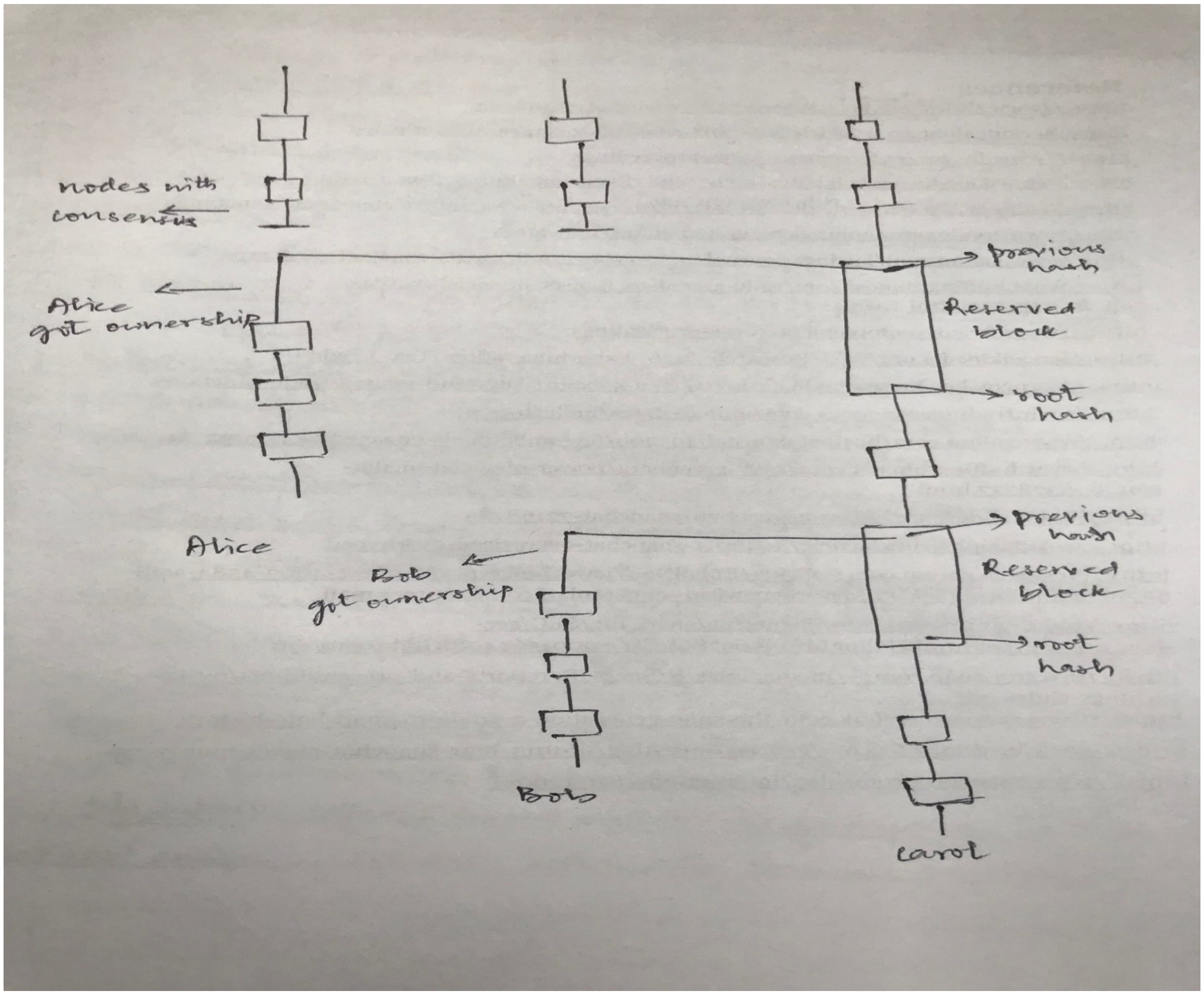

figure 2

- **Storage:** IPFS (Inter Planetary File System) is the popular decentralized storage service with its own sophisticated IPFS protocol. IPFS doesn't encrypt the data stored on the network (I was unaware of this until the meeting, apologies), users must do it by their end. Controlling the encryption and key management is in the hands of user. Secure encryption algorithms like GPG (GNU primary guard) can be used for data security.
- The popular files stored on the IPFS network have the longest shelf life as they are being distributed among many devices. The unpopular files can also be lost if the device holding the file goes down; We can run a cluster of nodes and make them into an IPFS swarm to keep the files permanently on the web. It would make it private & also redundant and encrypt the data to keep it protected from any malicious nodes.
- After encrypting the data; to securely delegate the access to the nodes and any users, the approach of **proxy re-encryption** can be used – which is being deployed and offered by a trustless decentralized service called NuCypher. This approach is better than **circular encryption** (apologies for being ignorant

about this technology during the meeting), in my opinion as mass delegation would be easier with the re-encryption approach.
- IPFS mobile clients are still in the build phase, but the mobile applications can utilize the power of IPFS using the IPFS Java APIs and JS wrappers are available to be used with IPFS. Blockchain support tools like Zippie and Nem enables the communication between mobile devices and blockchain.

Note: Nimses is a blockchain which follows the singularity approach, and they have a sophisticated mobile phone application which interacts with their blockchain. This is a hybrid system which uses the time stamp servers and distributed NewSQL DBs.

Summary and Broad System Design

1. **Standalone Blockchain / network:**
- This approach is more suitable for the system which has been discussed earlier as it would require to have a variable block size.
- Can be made agnostic to mobile devices and computers, (using ETH or BTC blockchain on mobile is still not an easy task, Geth lite for mobile devices is still evolving; no stable release yet). Timestamp server centralized approach or a decentralized timestamp tracking application can be designed.
- IPFS protocol can be leveraged to build a swarm of our devices, and natively built KMS system or a trustless decentralized KMS services like NuCypher can be used for the purpose of security.
- Summarizing from the earlier discussion – The nodes which are moving to the locations with low internet connectivity can take ownership and reserve a block. Once the ownership is decided the root hash can be provided based on the owner information and time of ownership delegation.
- The data collected from the node which are in low connectivity area and remote can push the data to the blocks assigned to them. They cannot add any new blocks until the current ownership is closed. The blocks are designed in the form of DHTs or Merkle trees (this is a bit hard approach).
- The keys to access of the data on the network, can be made available using the trustless distributed service NuCypher.
- This approach enables us to have a sophisticated application design ranging from cell phones to any kind of mobile devices.

2. **Using Ethereum network and services:**

- Token based system; by utilizing the Non Fungible Tokens – ERC721, using ERC20 doesn't fit as a proper solution.
- All the devices are given an ERC721 token which maintains the authenticity and uniqueness of the owner / node. Only the devices with right Eth address and right ERC721 token will have access to the data, adding data to the chain or retrieval.
- Geth lite version with JSON RPC wrappers such as JD APIs can be used to build smart phone applications, to communicate with the main network.
- The need as main net access is not necessary as this is used just for the purpose of authentication. Rest of the storage and block syncing still needs to be done on the systems. The Eth supporting services like side chains, bridges and oracles are not used.

References:

https://github.com/ethereum/goethereum/wiki/Mobile%3AIntro

duction

https://www.youtube.com/watch?v=R0Ia1U9Gxjg

https://ethereum.karalabe.com/talks/2016-devcon.html#19

https://nimses.com/whitepaper/

https://www.nucypher.com/

https://www.youtube.com/watch?v=hug

1LRznsQI

https://arxiv.org/abs/1707.06140

https://github.com/bpolania/Bianchett

o https://devpost.com/software/ipfs-

mobile https://ethereum-android.com/

https://medium.com/@mycoralhealth/learn-to-securely-share-files-on-the-blockchain-with-ipfs- 219ee47df54c