Aura Protocol: A Peer-to-Peer Blockchain Scaling Solution for Highly Interactive Decentralized Applications

Zhi Huang
Sam Snyder
Gabriel Schillinger

August 2018, version 2.0

Abstract

Aura is a next-generation high-performance public blockchain for interactive consumer applications. Its peer-to-peer verification technology allows for unlimited and sustainable scalability, and it provides a novel approach towards solving the blockchain scalability problem currently preventing blockchains from mass adoption.

1 Problems with classic blockchains

The scalability problem blockchains face today stem from the original application built with blockchain technology: Bitcoin.

Bitcoin was designed with a specific goal in mind: the secure exchange of value without a trusted third party. Its key innovation is **decentralized consensus** which removes the need for trust. On the other hand Bitcoin uses **replicated master ledger(s)** to store all account balances (in the form of UTXOs[1]). This is adequate for Bitcoin's purpose but very limiting. The replicated master ledger design is only adequate for services where:

1. Data access is unpredictable, i.e. any account can transact with any other account.
2. Transaction rates are low.

In other words, it's good for banking and financial applications but not good for games or chat apps.

The Aura protocol is designed specifically to support consumer applications with frequent user activity and peer-to-peer interactions, such as games and chat apps. Aura is able to scale because it makes it possible for every client to store different data and perform different computations.

Caveat: The Aura protocol requires predictable data access, for example game players interacting with objects and other players in its vicinity in the game; users chatting with others in the same group.

2 Scaling through Peer-to-Peer Network (P2P)

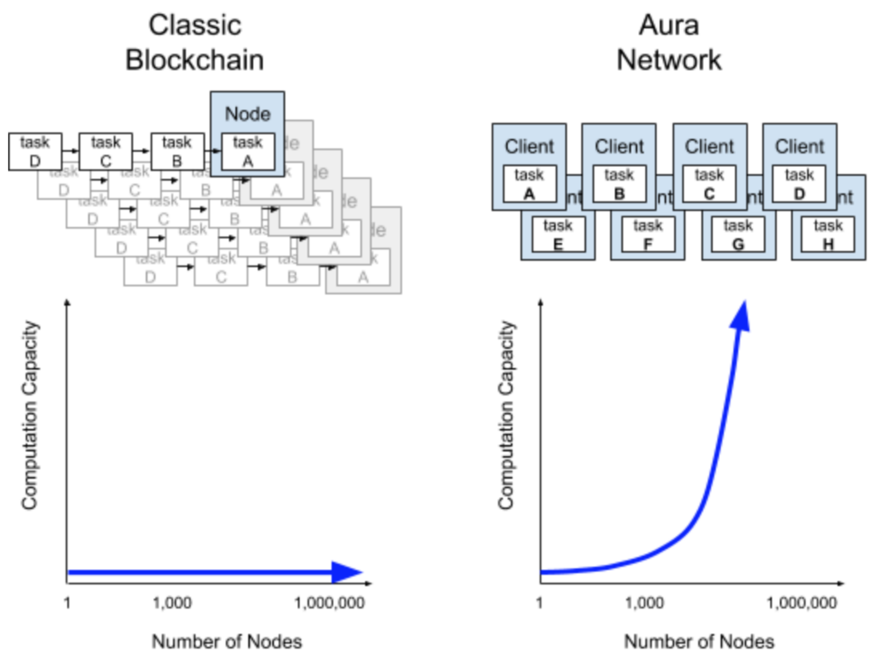
A key difference between Aura and classic blockchains is the location where application logic is executed and where data is stored.

Classic blockchains run DApp logic and store data on blockchain nodes. All nodes perform the same logic and store the same data, so the entire blockchain has the computational capacity of a single computer, no matter how many nodes exist on the network.

Aura runs application logic on the client network and allow clients to perform different computations and store different data. The capability of the network scales directly with the number of nodes. Each Blockchain Ledger can

[1]UTXO: Unspent Transaction Output is what can be used as input in a new transaction.

[2]based on the expectation that clients will on average commit once a minute, where each commit contains an average of 100 user transactions. The ledger is expected to handle 1000 commit per second.



manage tens of thousands of concurrent users and hundreds of thousands of transactions per second.[2]

2.1 Stateless Executor

Each Aura client is a stateless executor of application logic. In addition to processing its own user input and state data, each client can receive data from any other client and replicate their result. It is designed this way to support our P2P verification protocol.

2.2 Offchain Data Storage

A truly interactive application rich with user generated content will contain hundreds of terabytes if not petabytes of data. It is clearly not feasible to replicate all these data on each blockchain node. In developing Aura we realized that the hash of each data block is sufficient to guarantee data authenticity. Raw data can be stored off-chain.

On the Aura Network, owners of the data are responsible for their own data. This could mean using a cloud data store solution or keeping data on a private machine. Each user can choose what works best for them. When a 3rd party needs to verify data authenticity, it only needs to request data from the client and compare the state with the corresponding hash from the ledger.

3 Aura Architecture

3.1 Layer 1: P2P Network

Responsible for scalable compute, storage and content distribution. The P2P network can collectively handle hundreds of thousands of user transactions per second on the condition of predictable local data access.

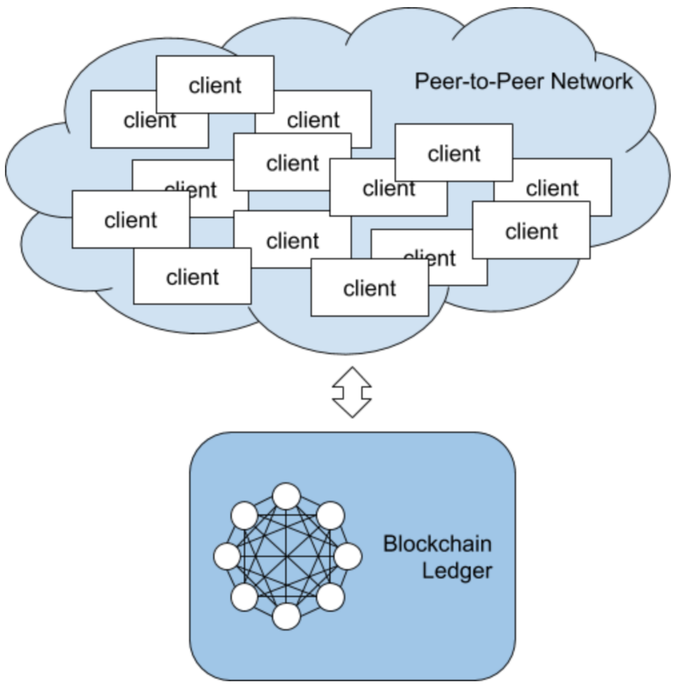
In a traditional blockchain, all nodes perform the same computation and store the same data, thus the capability of the network does not scale. The breakthrough of Aura is to push compute, storage and file transfer to the client network, allowing clients to perform different computations and store different data. This means the capability of a Aura network scales with the number of clients on the network.

3.2 Layer 2: Blockchain Ledger

The Blockchain Ledger plays a key role in the Aura P2P Verification Protocol:

1. It maintains a list of all online clients. Aura operates with the expectation that clients will go online and offline throughout the day. Clients notifies the Blockchain Ledger when they come online and maintain their online status by making periodical (every 1-5 minutes) commits.

2. It makes randomized peer verifier assignments. Client nodes check each other's computation, a result is only deemed



trustworthy when it has been verified by multiple (15 or more) peer clients. The Blockchain Ledger ensures each reviewer client is randomly selected from the list of all online clients to prevent any chance of collusion amongst malicious nodes.

4 Hash Function, RNG and Data Verification

The Aura protocol utilizes hash functions in a number of key places to ensure cryptographic security of the network. This include random number generation (RNG) and data verification.

4.1 Definitions

A **hash function family** $H = \{H_K\}_{K \in K}$ is a function $H : K \times D \to R$ where $|D| < |R|$. We generally view K as a probability distribution on the set of possible keys but here we also use K to denote the set of possible keys.

A hash function family is said to be **Universal** if and only if for all $x_1 \neq x_2 \in D$,

$Pr[HK(x_1) = HK(x_2)|K \leftarrow K] = \frac{1}{|R|}$,

and thus being a universal hash function family is equivalent to having a probability distribution on functions from D to R that maps elements of D in a uniform pairwise independent fashion.

Universal One-Way Hash Function Families (UOWHFFs) satisfies the condition where given the key K and a challenge point x_1, it is hard to find a second point x_2 that maps to the same place as x_1. It is also known as target collision resistance.

Algorithm 1 SHA256

```
1: function SHA256K($M$)
2:     $y \leftarrow$pad$(M) = M10^d l$ where $l$ is the 64-bit representation of $|M|$ and $d$ is chosen so that $|y|$ is a multiple of 512.
3:     Write $y = M_1M_2...M_n$ where $|M_i| = 512$.
4:     $V_0 \leftarrow IV \in \{0,1\}^{160}$
5:     for $i = 1$ to $n$ do
6:         $V_i \leftarrow$shf1$_K(M_i, V_{i-1})$
7:     end for
8:     return $V_n$
9: end function
```

We select the SHA256 hash function for its universal one way property and collision resistance.

4.2 Deterministic RNG

Since a universal hash function produces a uniform pairwise independent mapping between its input space to its output space, it can be used to generate random numbers of uniform distribution. Furthermore, since a hash function always produce the same output for the same input, the random number can be independently reproduced by knowing the input data.

The Aura Blockchain Ledger produces a new block ever second, the root hash of each block is a SHA256 hash of the data in that block. The root hash is uniformly distributed and impossible to predict ahead of time.

Combining the root hash of a block with another piece of data, such as client account ID, we can produce a random number for a specific client on a specific block. This is used to select peer verifiers.

4.3 Data Verification

A property of a Universal One-Way Hash Function Family such as the SHA256 is that the chance of 2 different inputs producing the same hash is the inverse of the total number of possible hashes. For SHA256 that number is $1/2^{256}$.

To ensure the integrity of a dataset, no matter how large or small, one only need to store the 160 bit SHA256 hash of that dataset on the blockchain. Anyone can verify the authenticity of an off-chain dataset by comparing it's hash with the one stored on-chain.

4.4 Future Upgrades

Security of Aura network can be further improved, at cost of additional computation by adopting the SHA3 standard.

5 P2P Verification

The Aura Peer-to-Peer Verification protocol is designed to safeguard compute and data integrity and does so in a highly scalable fashion.

5.1 Client transaction processing and snapshot commits

Transactions are processed on the client side and update the application state locally. Each client node periodically takes a snapshot of its application state, computes its SHA256 hash and tries to commit the hash to the blockchain ledger.

5.2 Delegated Transaction Verification

The Ledger does not verify client transactions directly, it delegates this task to other client nodes (verifiers) in the network. The blockchain ledger assigns a verifier client to re-run transactions in order to verify the submitted result.

5.3 Verifier Selection

The blockchain ledger maintains a collection of all online clients. A client registers itself with the ledger when it comes online. The ledger stores the client, using the hash of the client identification as the key. This uniformly distributes all clients in the search space.

When a client requests a verifier, the ledger generates a random 'lookup key' by computing the hash of the combined data of requesting client's identification and the root hash of the current block. The verifier is found by searching through the collection of online clients and finding the largest key smaller than the lookup key.

The uniform distribution of both client keys and the lookup key in the search space means all online clients stands an equal chance of been selected as verifier. The randomness in the lookup key, due to the inclusion of the current block hash, makes verifier selection unpredictable beyond the current block.

5.4 Lookback Verification

An assigned verifier not only checks the current commit, it will also check the last N commits from the same client. N is typically 15 but can be adjusted, a higher N makes it harder for a client to cheat but increases the overall computation overhead of the P2P network.

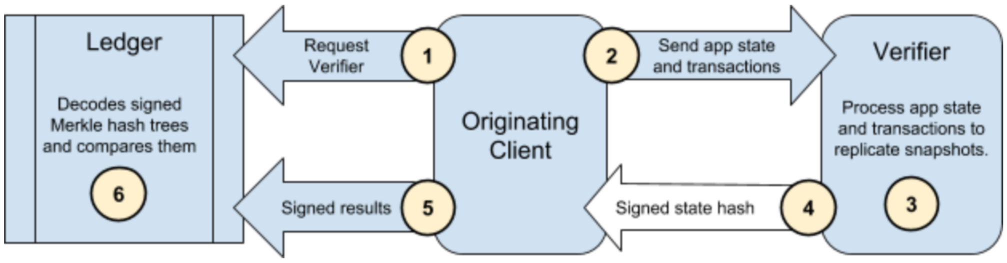
The originating client must submit to the verifier its state from N commits ago as well as all subsequent transactions. The ledger also need to store hashes of the N most recent commits in order to support this verification.

Lookback commit verification is key to ensuring client computation integrity. In order for a client to temper with a commit and not be caught, it needs to control all N verifiers in subsequent commits. The probability of this is exceedingly small.

The verifiers are randomly selected and cannot be predicted ahead of time. Even if the attacker controls 50% of client nodes, the chance of all N verifiers to be under the attacker's control is $1/2^N$, for $N = 15$ this is 1 in 32 thousand, if $N = 20$ this is 1 in 1 million.

It only takes a single honest verifier to raise the alarm and trigger arbitration by the blockchain ledger. Any tampering will be found and all clients involved in the attack will be suspended or banned.

An attacker may try to repeatedly request verifiers until one of the nodes it controls is assigned. This is why a client must submit a commit request once a verifier has been assigned. A client is allowed to re-request verifier once every $N + 1$ commits, exceeding this limit will cause the blockchain ledger to verifier client history directly. Even if no fault is found the client will be fined for the additional work it has incurred.


Ledger
Decodes signed Merkle hash trees and compares them
6
Request Verifier
1
Originating Client
2
Send app state and transactions
Verifier
Process app state and transactions to replicate snapshots.
3
Signed results
5
Signed state hash
4

5.5 Staking - Sybil Attack Prevention

A sybil attack is when an attacker is able to create and control a large number of nodes on a blockchain network, compromising its security. The Aura P2P network uses staking to prevent sybil attacks.

Each client account is required to deposit a small amount of Aura tokens before they are able to participate in the verification process and earn fees. The deposit is locked up for a minimum of 30 days and can be confiscated if the client is caught attacking the network.

Staking makes it very expensive and risky for an attacker to create and maintain a large number of fake nodes on the network.

5.6 Confirmation Time

Due to lookback verification the trustworthiness of a commit increases with each subsequent commit until there are N subsequent commits. This is conceptually similar to confirmation time in Bitcoin, a high value transaction should wait for more confirmations than low value ones.

A client can speed up confirmation time by making more frequent commits at the cost of more fees. Arbitration by Blockchain Ledger Each blockchain node can also run client logic. Whenever there is a mismatch between verifier result and the submitted commit each blockchain node will verify results directly and come to a consensus of the true outcome. This way disputes are always resolved.

5.7 Punishment and Deterrence

Client nodes are considered to have colluded with an attacker if they failed to flag an incorrect commit, i.e. returning a false positive verification result. Any client nodes caught in this manner will be severely punished, resulting in account suspension or a permanent ban.

On the other hand, clients that dispute snapshots by mistake (i.e. false negative) will be treated more leniently with only a small fine to cover the cost of arbitration. This encourages verifiers to be more aggressive in flagging potential issues.

5.8 Sequence Diagram and Code Example

Code Example 1: Simplified client function for preparing and submitting state hash

```
function commit(txBlocks, previousStates, currentState) {
  // 1. Get assigned verifiers from blockchain ledger
  LedgerAPI.getVerifier(this.accountId)
  .then(reviewer => {
    // 2. Send transactions and last state verifier
    return peer.requestVerification(txBlocks, previousStates[0])
  })
  .then(signedResults => {
    // 5. Commit to blockchain ledger
    const statesHash = SHA256.digest([startState, endState])
    LedgerAPI.commit(statesHash, signedResults)
  })
}
```

Code Example 2: Client function to verify transactions and sign the result

```
function verifyTxBatch(txBlocks, startState, callback) {
  // 3. Process app state and transactions
  const appEngine = new AppEngine(startState)
  const states = txBlocks.map(txs => {
    appEngine.process(tx)
    Return appEngine.getState()
  })
  states.unshift(startState)
  // 4. Sign verification result
  const statesHash = SHA256.digest(states)
  const result = secp256k1.sign(statesHash, this.privateKey)
  callback(result)
}
```

6 Blockchain Ledger

Aura's Blockchain Ledger act as coordinator and authority to the P2P network. It runs on the Byzantine Fault Tolerant (BFT) Delegated Proof of Stake (DPOS) consensus model with a known set of block producers.

The Aura Ledger is a specialized blockchain with specific functionalities designed to coordinate the P2P client network. It does not run general purpose VMs because application logic runs in the P2P client network. This specialization allows Aura Ledgers to be optimized for maximum transaction throughput.

6.1 User Asset Ownership and Trading

Digital assets within the application are whole represented by the app state which is managed by the client.

In order to trade a digital asset with another user, the digital asset must be first extracted from the app state and placed on the ledger. The user needs to generate a 'deposit' event which takes items out of the standard app state section and into a special deposit section. Any items placed in the special deposit section will be deleted after N commits, enforced by the verifiers. On the Nth commit, the client will send proof to the ledger of the item been in deposit for N commits and it will be placed in the client's account on the ledger.

When transfering digital asset from ledger to local app state, the user need to first release the item on the ledger then add it to local state. Verifiers will check the release and the local state to ensure there is no double spending.

6.2 Parallelization and Sharding

Aura's primary innovation is scaling through a peer-to-peer architecture. Aura network can be further scaled through parallelization and sharding.

Different applications needs different sharding schemes. For games a player's interactions with the world and other players are highly localized. Games can partition the client network along in-game location boundaries. Densely populated areas can be subdivided further.

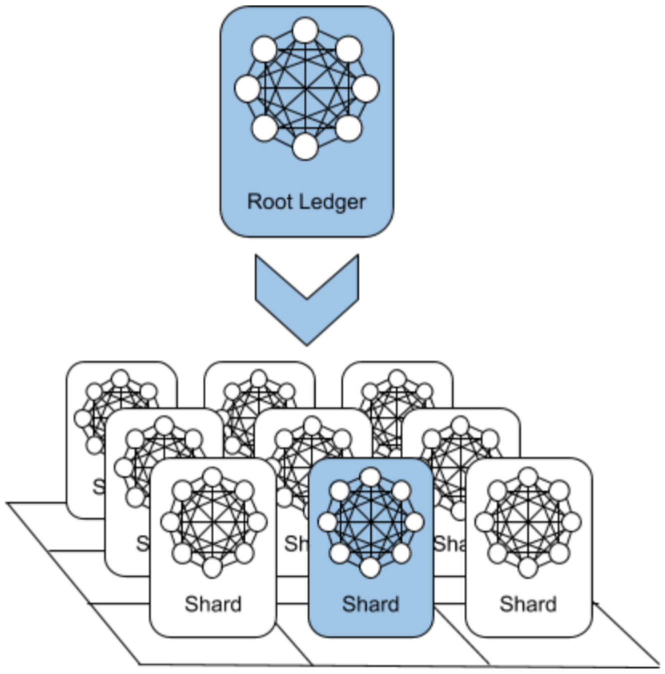
7 Conclusion

Aura Protocol is a high-performance public blockchain where computation and storage takes place off chain on the application client. A peer-to-peer verification protocol is developed to guarantee integrity of computation and data.

General purpose block chains are most often ideal for registering and tracking critical data to their ledgers—identity, currency, and tokenized assets are the most common instances. Structural optimizations beyond the capabilities of general purpose chains are required to meet the demands of highly interactive decentralized application and the varied economic and social activity within them.

By its nature, the Aura Protocol enables anyone to launch a decentralized application of their own. In the same way that any ethereum user with a wallet address can deploy a smart contract to the ethereum network, so to will any Aura user be able to launch their own decentralized application.

The first use case of the Aura Protocol will be powering the Aura Network, a decentralized games and virtual worlds platform. Applications of blockchain technology in games in-



clude using on-blockchain digital assets to represent custom currencies and digital assets (in-game currency), the virtualization of an underlying physical object (tethered-parallel universe), non-fungible assets such as virtual property, as well as more complex applications involving digital assets being directly controlled by a piece of code implementing arbitrary rules (smart contracts) or even blockchain-based decentralized autonomous worlds (DAWs). What the Aura Network provides is a games-logic protocol with a built-in fully scalable p2p networking scheme and sharded ledger structure that can be used to spawn and dynamically scale DGames and virtual worlds, allowing developers to create any of the systems described above, as well as many others that we have not yet imagined, simply by writing up the logic in a few lines of code.

References

1. Plasma White Paper `https://plasma.io/plasma.pdf`
2. TrueBit White Paper `https://people.cs.uchicago.edu/~teutsch/papers/truebit.pdf`
3. Ethereum White Paper `https://github.com/ethereum/wiki/wiki/White-Paper`
4. Raiden Network Documentation `https://github.com/raiden-network/microraiden`
5. Casper `https://blog.ethereum.org/2015/08/01/introducing-casper-friendly-ghost/`
6. Storj `https://storj.io/storj.pdf`
7. SONM `https://sonm.io/Sonm1.pdf`
8. Swarm `https://github.com/ethereum/wiki/wiki/Swarm-Hash`