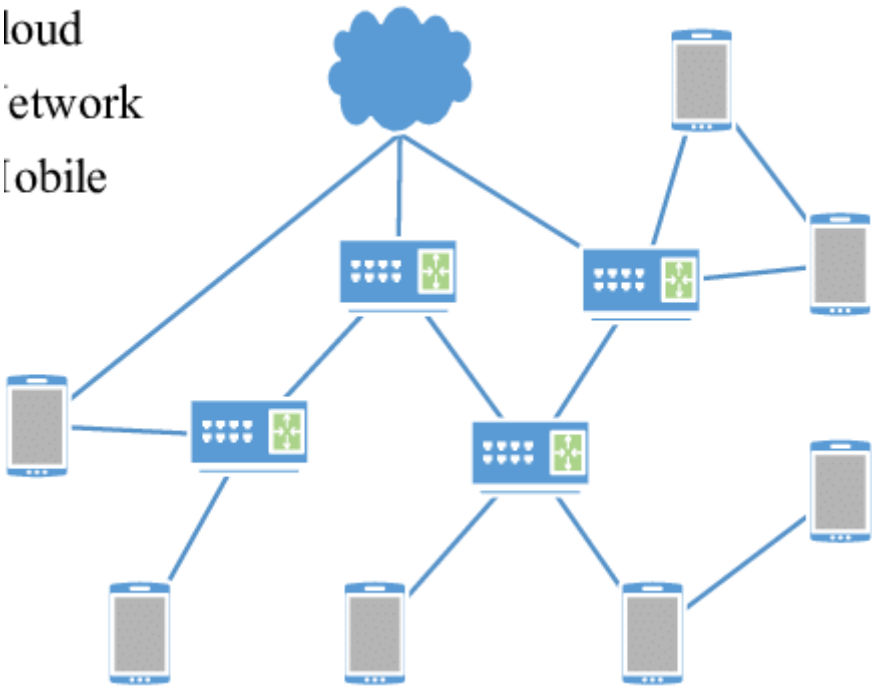
Dylan Follow
6 min read · Draft · Listen

How Edge Computing enables Decentralization



Hi everyone. Today I wanted to clear up some confusion around what defines **Edge Computing**, **Serverless Architecture** and how it fits in to **Decentralization** and **Web 3.0** Applications.

Blockchain and Decentralized Applications are the new craze, but there's one problem. These networks and Apps have scalability issues, relying on centralized authorities and cloud providers like Amazon Web Services, Google Cloud, Azure and Private Clouds, to power their applications and workloads on Virtual Machines, Containers and Servers sitting in central data centers somewhere. You cannot simply host an app on the

as a Single Computer. We have seen projects like **Internet Computer Protocol** kick off and play with this idea, but even then… True trustless and Decentralized Architecture is yet to be implemented. ICP simply partners with "trusted" data centers to separate transactional workloads on chain and application specific workloads onto this set of data centers. This leaves out average users and the common public to have stake in providing compute resources in a trustless, anonymous way. How is Edge relevant to all this?

Edge Computing is simply a paradigm that enables Operators and Developers to offload Workloads that are concerned with Real-Time Data. By bringing a set of inter-connected devices that process workloads and functions physically closer to the end-user, nodes are distributed across world-wide regions without a single point of failure. This is because the collective set of nodes create a service mesh that can indefinitely scale based on the volume of edge devices.

Just to simplify, I will use the words "devices", "servers", "nodes", "edge nodes" to describe the same thing.

Edge Computing enables Serverless Architecture by allowing end users to only focus on applications and functions without needing to manage their own infrastructure. Just to clarify, Serverless still requires "servers", in this case the edge nodes act as "servers", this is semi-decentralization because a generic DNS or ENS (Ethereum Naming Service) endpoint will invoke a request on any given node based on criteria (location, optimization requirements).

Now you may ask: How does this fit into Web 3.0?

Well, what I envision is a cross-chain (Bitcoin, Ethereum, etc) protocol that leverages Verifiable Random Functions (VRF) for establishing peer-to-peer connections between edge nodes and end-users. In this architecture, users are incentivized to provide computing space on their idle devices, browser tabs to collectively create a distributed, decentralized edge network. With current technologies including Oracles (Not the company Oracle…We will explain later), WebRTC, STUN, IPFS, JSON-RPC, Software Guard Extension (SGX), Distributed Hash Tables (DHT) and WebAssembly (WASM), it is

Much like the TOR Protocol, Bit Torrent, and Blockchains, this network uses peer-to-peer connections to proxy and route traffic to random nodes in a given region or set of edge nodes. The difference is that common strategies such as UDP firewall hole punching to facilitate connections within a Distributed Hash Table would be implemented with Blockchains to provide verifiable transparent randomness, and governance over everyone.

Much like an Onion, at the core is simply more layers, represented as Edge devices. Edge devices in this context can be ANY device that is interoperable with the web and HTTP. Different types of responsibilities are assigned to nodes in order to ensure Anonymity, Trustless Execution and Interoperability with different Ecosystems. Every node can perform these three objectives kind of like Polymorphism in Object-Oriented-Programming. A key and value pair represent a node and the information needed to connect to it, so if an end user invokes a Serverless function, a set of multiple nodes will be randomly selected to either individually or collectively execute the function. Multi-party computing is the idea where multiple nodes can act as a single server, relaying processes to each other and taking turns executing pieces of the function. Think of each node almost as an individual CPU core or Hardware function that can collectively combine their resources to complete a single task.

> A primitive way to measure this test of zero-knowledge execution is with a simple Dot Product. (An operation that will compile two equal length sequences of numbers into a single number) Let x & y; be Vector 1 and Vector 2 where N can be the dimension of vector space and x of i; and y; of i; represent a specific iteration through the Summative Notation where i; can represent any value... lets just say 1.




values and return the same output synchronously; without being aware of each others variables (via SGX or TEE) , then that can provide a simple mathematical proof in between individual execution processes and time stamps of a given Serverless Function.

$$a \cdot b = \sum_{i=1}^{n} a_i b_i$$

What is an Oracle?

A Blockchain Oracle is simply an off chain service (Ideally hosted on a peer-to-peer edge network) that bridges Real-Time off chain data in a way that Smart Contracts and Blockchains can consume. Each query to an Oracle creates an event-on-chain in order to invoke a Smart Contract, Transaction or any Blockchain Event. Oracles have many use cases, An example is in providing Stock Prices for a trading dApp and invoking Buy/Sell Orders through smart contracts based on current prices. Oracle technologies also enable Smart Contracts to invoke API Calls, HTTP requests and any External interaction with the Real-World based on a Contracts given Criteria. This eliminates the need for a middle man, because a trusted data source is being used to computationally and mathematically guarantee SLAs, Contracts and Agreements.

Oracles are important in this context because this is how the edge network would implement a form of trustless governance over the state of the network, regardless of identity, intention (hosting a node or using a node for compute) by relaying signatures, proofs and minimum execution information to a set a on-chain validator nodes in order to record and publicize Edge Network events and applications being run. Users would be




also strongly incentivizes users to validate events for rewards and bounties issued autonomously by the Edge Protocol.

Challenges

The first challenge of ZKP mining is the acceleration, this has been covered in Paradigm's article. Paradigm promotes the idea that FPGA is better than GPU and ASIC for given hardware acceleration, I have differing opinions. I believe that GPUs are still be the mainstream hardware for ZKP computing in the foreseeable future due to:

1. ZKP algorithms will frequently change in the future, and GPUs are the most developer-friendly hardware.
2. Mining rigs hold a large number of GPUs, especially for the Ethereum community. When Ethereum finally switches to PoS, these GPUs will be available for ZKP mining.

The second challenge is the parallel computing of ZKP on distributed hardware. Currently, there are three types of solutions:

1. If each proof takes only a little bit of computation, such as Aleo, the traditional mining pool is suitable for assigning computing tasks to distributed hardware.
2. Sometimes we can separate a single proof into multiple proofs, such as zkSync and Filecoin. In this case, the separated proofs can be computed parallelly and then aggregated into a single one.
3. DIZK is an architecture able to compute a single proof in a distributed manner.