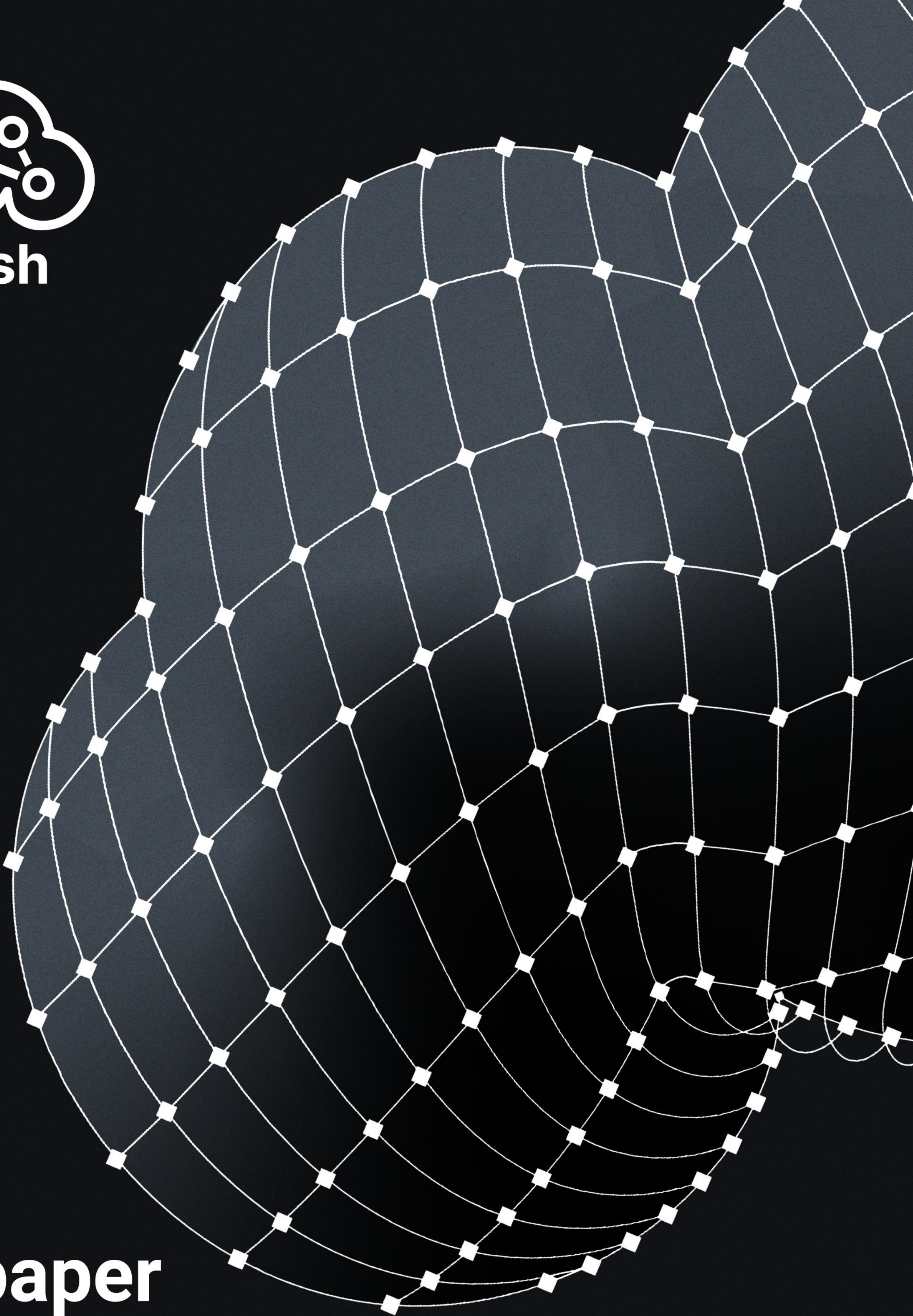
moosh



Litepaper

Version 1.0
September 2024

moosh.gg

Moosh Net

Moosh Team

A decentralized social network for joining trustless &
trust-dependent systems and sharing digital resources

September 14, 2024

Contents

1. Introduction 3
  1.1. Foreword 3
  1.2. Why Wasn't Web3 Built Sooner? 4
  1.3. What is Web3? (The Pillars & Pancakes of Web3) 5
    1.3.1. Web3 Structure - The Horizontal Stack 5
    1.3.2. Web3 Structure - The Vertical Stack 7
  1.4. The Problem of Joining Web3 to Web2 - Don't Trust, Verify! 9
  1.5. What is Web3 Missing? 9
2. Moosh - The Design Blueprint 12
  2.1. The Philosophy Matters 12
  2.2. What Moosh Fixes 12
    2.2.1. A Reputation Network for Digital Social Systems & Commerce 12
    2.2.2. Decentralized Cloud Computing 12
  2.3. Building Moosh 13
    2.3.1. Resource Sharing Marketplace 14
    2.3.2. Social Graph & Overlay Mesh Network 15
    2.3.3. Verifiable Compute Engine (VCE) 16
3. Minimum Viable Product (MVP) 17
  3.1. Clients and Providers 17
    3.1.1. Virtual Machine Manager 17
    3.1.2. Resource Provider Inventory 17
    3.1.3. Nix to Manage VMs 18
  3.2. Initial Use Cases - Bootstrapping Network Adoption 18
    3.2.1. Bridges 18
    3.2.2. Link-based Invite System 20
    3.2.3. Incentivizing Utilization of Spare Computers 20
    3.2.4. Fully Programmable Provisioning (FPP) 21
    3.2.5. Potential Future Use-cases 21
  3.3. Tokenomics 22
4. Contact Information 23
5. Acknowledgements 24

1. Introduction

This is the v1.0 litepaper for the Moosh Network, a.k.a MooshNet, a.k.a Moosh.

Moosh is a decentralized peer-to-peer (P2P) social network for joining trustless & trust-dependent systems and sharing digital resources. We believe that upgrading today's centralized Internet (Web2) is long overdue. For this new Internet (Web3) to truly take form, the world must merge trustless electronic systems with trust-dependent human systems. Today, these don't connect well.

> Moosh : A term of endearment for a trusted partner

> Moosh : Mesh Topology + 2x (Open Overlay) = 'Mesh' + 'oo'= 'Moosh'

TLDR: Moosh exists because Web3 is not yet adequately crafted. The necessary technologies to build it already exist, and the wait has been long enough.

1.1. Foreword

Computers in the 1960′s were large and immobile. In order to make good use of information stored in any one computer, the user had to travel to the site of the computer or have magnetic tapes sent through the postal system.

In the 1980′s new communications protocols were established. Transfer Control Protocol and Internet Protocol (TCP/IP) allowed different types of computers on different networks to communicate via a universal standard. This was Web1, connections for read-only data. The vast majority of its users only consumed content, never publishing.

The Web2 Internet was termed in the 2000′s when software applications began to be built upon on the Web as opposed to upon a computer's local desktop. Web2 allowed users to both consume and publish data over the network. This saw the advent of user-generated content, a focus on ease of use, and spawning of collaborative tools.

Both of these Internets originated as a new and improved means to communicate; this was their foundational property. Only afterwards did e-commerce, social media, and other paradigms arise. To be sustainable, the business models of these emergent enterprises rely on commercialization of user data at the cost of user privacy.

Enter Web3. What is it's foundational property?

Web3 maintains the read and write functions of shared data but adds the concept of ownership. Moreover, ownership of data and privacy is returned to the end user as opposed to the service provider. With this come the downsides of increased responsibility and reduced convenience. The upside being the users are wholly in control of their data and online identity.

The theme of decentralization is core to the philosophy of this third iteration of the Internet.

Web3 has yet to take a true decentralized form. It's a work in progress. Before the world builds or sells use cases for Web3, let alone expects mass adoption, the effort to build a new and indispensable decentralized fabric of the Internet must be completed.

Web3 can't expect mass adoption if it can't at least do what Web2 does, plus more. Web3 is absent of the power of its Web2 predecessor, for example it has lackluster big data or AI capabilities. Drawing analogy to Web2, Web3 is in its dial-up era right now and we need fiber.

What is the fiber of Web3? It's Decentralized Physical Infrastructure Networks (DePIN).

1.2. Why Wasn't Web3 Built Sooner?

Building Web3 *properly* wasn't focused on until recently. Web3 needs more than blockchain alone, but it's an important part of the puzzle. Historically, main attention was drawn to blockchain's application to currency - cryptocurrency. In 2024 we are in the fourth market cycle of the cryptocurrency world:

- Cycle 1 (1980-2010): Cypherpunks (Ecash°, Bitcoin°)

  'We are clever, look at this complex thing we built' - Very few people

- Cycle 2:(2010-2017): Libertarians

  'We will fight for Freedom of money' - A few people

- Cycle 3 (2017-2021): Speculators (ICO craze, DeFi Summer)

  'We can get rich quick.' - Many people

- Cycle 4 (2022+): Global Stage (ETFs, TradFi & Corporates Enter)

  'The world takes notice, finally.' - Mainstream

Trading and speculation were much higher margin, more profitable, than building new infrastructure. It wasn't widely apparent to it's adopters that blockchain had more use cases than money, but where this was understood those sectors of the ecosystem were rife with vaporware, e.g. during the Initial Coin Offering (ICO) craze of 2017. This era left a bad taste in investors' mouths, making it difficult for Web3 infrastructure founders and builders to source funding and attention.

It was mentioned that Web3 is more than just blockchain. Learn more about what it's made of in the next section.

1.3. What is Web3? (The Pillars & Pancakes of Web3)

It is essential to understand the anatomy of Web3 before explaining what is missing, i.e. how great is today's shortfall versus a 'true Web3?'

The structure of Web3 can be visualized using two compositional models:

1. A horizontal stack - 'Pillars'
2. A vertical stack - 'Pancakes'

Both are explained below.

1.3.1. Web3 Structure - The Horizontal Stack

The first model compares Web3 to a strong stone building comprised of three pillars and one heavy roof.

The pillars represent three critical components of Web3: Data, Infrastructure, and People. Collectively, these supports hold up the roof which represents Web3. If any of the three pillars collapse, there is insufficient total structural support for the roof, causing the roof to fall.

This model illustrates a necessity for the industry to target a full stack redevelopment for Web3 where all three critical elements (Data, Infrastructure, and People) embrace the philosophy of decentralization. Decentralization increases transparency, resilience and autonomy. It also reduces censorship, manipulation risks, and single points of failure.

A pillar fails if it is not decentralized because any point of centralization in Web3 is a weakness and at worst an attack vector for its downfall.

Why are these three elements the critical pieces of the puzzle?

- **Data** is arguably one of the most valuable resources in the modern world. Data fuels applications, without the flow of data into applications, they aren't fed, ceasing output.
- **Infrastructure** encapsulates hardware, virtualization, storage, and network. Similar to how your local computer needed compute resources (CPU & GPU), storage, and memory, so does the cloud. These ingredients are essential to run services, services which are software applications to which data is fed. Anything from short messaging services (SMS) to trading exchanges (CEX, DEX), applications require hosting, they need machines to render interfaces and compute with great quantity.
- **People** control the governance and development of technology, projects and businesses. They build and lead the progression of these ventures, driving them forward as builders and leaders but also as advocates and adopters of the technology. Any Web3 project is defined by it's community, including the security of any associated protocols.

Each pillar fails if it is centralized. To be specific:

- **Data**: Data must always be obtained by means of software intermediaries. The streams controlled by these intermediaries can be rate-limited, or switched off entirely by their centralized provider. Furthermore, these providers, having ultimate control and limited bandwidth, often paywall access to these streams. Ultimately, if an application owner can no longer afford to pay for the data, is banned, or is rate limited by these providers; their application is starved and rendered non-functional.
- **Infrastructure**: If centralized businesses control the machines which serve an application then the machine(s) can be isolated and taken offline by these parties. Furthermore, because public cloud services are hosted by third-party providers, secondary concerns can include: less control over underlying cloud infrastructure, data privacy, integration complexity with existing systems, unforeseen costs and unexpected expenses e.g. egress[1]. All of which can contribute to risk of vendor lock-in.
- **People**: Traditional business structures pose centralized choke points for services offered to the public by projects and companies. There is also key-person risk from small teams and from small leadership groups who are part of a wider community - those with concentrated or overriding control of community projects.

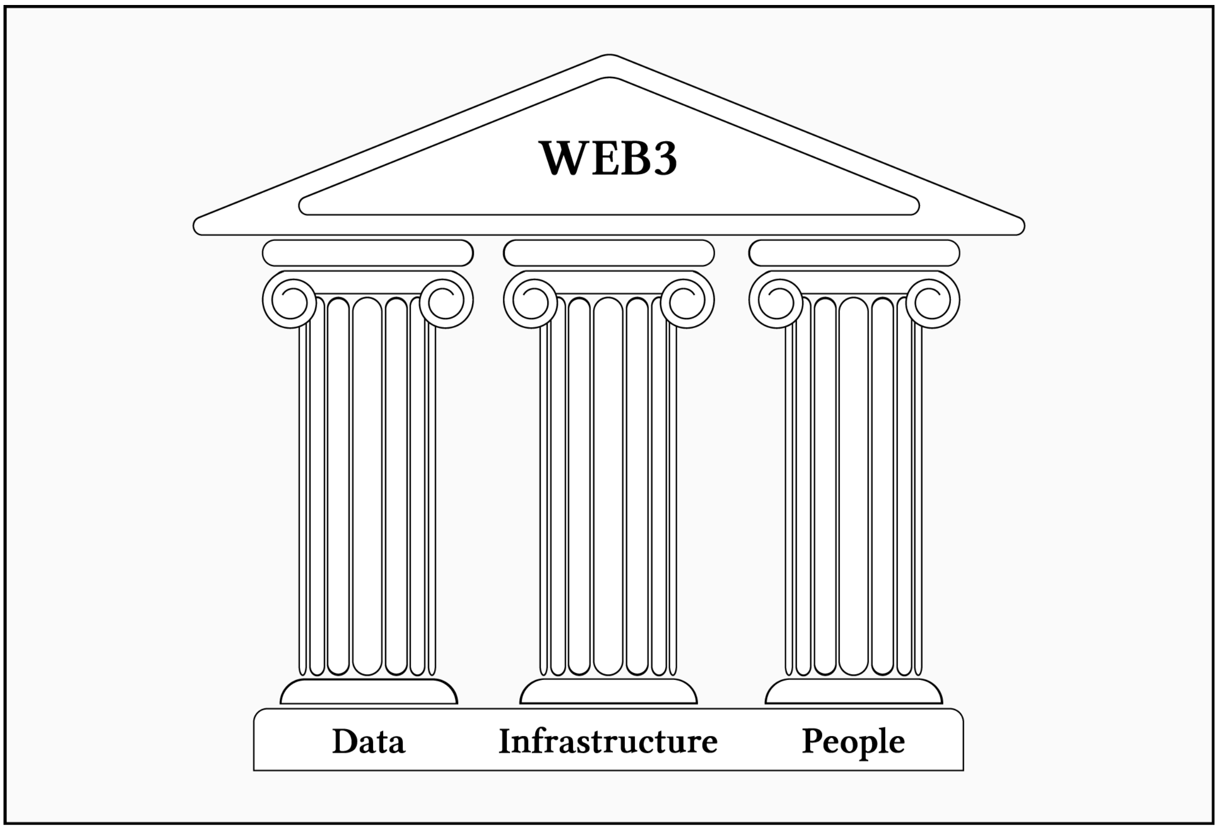


Figure 1: Horizontal pillar structure of Web3. Without real decentralization of each core component (Data, Infrastructure, and People) Web3 has one or more vulnerabilities.

[1]Data egress fees are cloud provider charges for relocating or transferring data from the cloud storage where it was first uploaded.

1.3.2. Web3 Structure - The Vertical Stack

The second model builds Web3 from a bottom-up approach. Beginning with the bottom layer and ending with the top, the three layers are:

1. Transactional Layer

   This is made of the blockchains most readers of this paper will be familiar with (e.g. Bitcoin and Ethereum). There are further sub-layers such as layer two's (L2s), sidechains, parachains, rollups… and all kinds of interoperability solutions. But collectively this can be simplified as a transactional layer (tx layer).

   This layer consists of two types of participants: (1) humans controlling crypto wallets, and (2) code controlling wallets (smart contracts). They store and exchange things of value. Participants may send coins and tokens between each other, send other data between each other, collaboratively store data in a decentralized manner e.g. with the InterPlanetary File System IPFS°, or transact together in other unique economies via defined protocols.

2. Application Layer

   The transaction layer feeds up to the application layer. Here Decentralized Applications (DApps) such as Decentralized Exchanges[2] (DEXs), NFT Marketplaces[3], and Web3 Games thrive.

   They connect to the transaction layer to ingest it's data and in return write new data. For example, a DEX will sync blockchain transactions to create a digital ledger of wallet balances and trading positions and provide a frontend to interact with on-chain liquidity pools and Automated Market Makers (AMMs).

   The application layer is synonymous with the 'access' layer, a portal by which to interact with the raw underlying transaction layer in a more accessible and meaningful way.

   Another leading example here are crypto-wallets[4] taking the form of web-based browser extensions or mobile device applications.

   In general, this layer offers users methods of interacting with the transaction layer by combining individual transactions or pools of transactions, and also interprets the tx layer's data to display it in human-readable formats (e.g Etherscan° or Dune°).

3. Identity Layer

   Typically in Web2 a user needs an account before interacting with an application. E.g. an email address is a common prerequisite before being able to open a social media account or online shopping cart. These applications wants to know who you are - they want an identifier.

   In Web3 this is the same… but also different.

   Here, your identity is not only analogous to the unique identifier of an email address, but it simultaneously serves a second purpose as your location of storing financial assets (and other data of value).

In Web2 your identity (email, passport, driver's licence, social security number) and bank account details are separate. In Web3 your public key (wallet address) and private key pair combine to serve both purposes. This poses benefit and hindrance depending on your holistic user persona for intended wallet use.

Note this is an introductory simplification of a larger multi-layer system, e.g. networking and protocol layers are omitted.

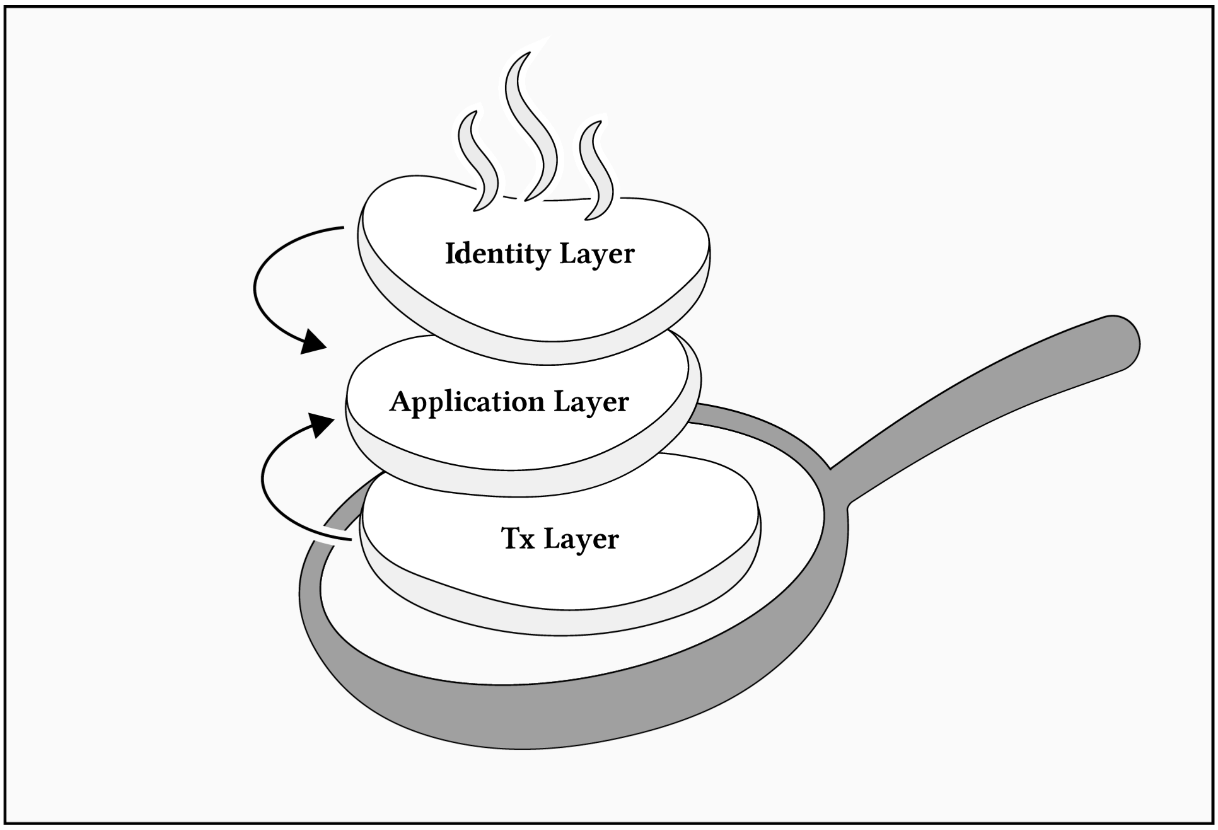


Figure 2: Vertical stack of Web3 conceptual layers. Immutable blockchain data from the transactional layer feeds the application layer above. From above, identities of agents are used as (a) logins to access applications as users, and (b) as unique addressable accounts for which to send & receive data transactions to correct destinations.

[2]E.g. Uniswap° Jupiter°
[3]E.g. Magic Eden° Blur° Opensea°
[4]E.g. Metamask° Coinbase Wallet°

1.4. The Problem of Joining Web3 to Web2 - Don't Trust, Verify!

How does Web3 interface with the existing Web2?

In general, there is a need for solutions to join Web3 trustless digital systems with trust-dependent Web2 digital and human systems. This is the *oracle problem*. Oracles are the current infrastructure solution to bridge between the blockchain (on-chain) and the outside world (off-chain). For example, those built by Chainlink°. This problem relates to the *Tx layer* of the pancake model and *Data* pillar in the pillar model because both rely on blockchain technology.

Blockchains have no built-in functionality to pull in data from any external system (e.g. one from Web2) or push data to them. They are therefore isolated networks and this sheltering is analogous to a computer with no internet connection, its internal state has no effect or knowledge of the outside world.

Consider these three examples:

- A Web3 application in the logistics space, supply chains require external verification that goods are of satisfactory quality and quantity at each step in the processes they monitor.
- A trade finance application requires human confirmation that trade documents were exchanged or relies on digital signatures from Web2 electronic systems.
- Asset-backed stablecoins rely on a guarantor to attest the digital asset is backed at minimum 1-to-1 with it's traditional finance equivalent.

Expanding on this final example, if any Real World Assets (RWAs) are to be represented on-chain via a digital twin, at some point in time a traditional electronic system or human(s) need to vouch for its Web3 equivalent's value, state, and/or exchangeability. Zero of this prerequisite information is internally generated within blockchain, nor are connections to the necessary Web2 or human services directly accessible.

On a positive note, blockchains being isolated networks allows them to be tamper-proof and be able to guarantee reproducibility and immutability because there are no external dependencies. This is what makes blockchains trust-less, in the famous words of Bitcoin champion Michael Saylor: 'Don't trust, verify!'

1.5. What is Web3 Missing?

Partial solutions to Web3 exist but are fragmented. Below is a summary of some notable deficits.

1. **Decentralization at the fundamental level has not occurred**

   It is inconvenient an uneconomical for users to purchase and maintain personal hardware. Out-of-reach hardware specifications for numerous popular blockchain nodes running the proof of stake consensus mechanism which are beyond the typical capabilities of a household machine. Public cloud offers cost-savings and consequently a large portion of blockchain nodes supporting the foundational tx layer of Web3 reside on centralized servers - i.e. this tx layer is not decentralized. As a result the data feed to DApps is at risk of censorship, tampering, and/or shutdown.

Assuming the above situation was remedied with perfect decentralization achieved for key blockchains' nodes, there is still an issue. The DApps living on the application layer run on centralized servers, applications are subject to the same risks as the aforementioned nodes.

To fix this common issue for both layers, the machines depended on need to (a) be decentralized and (b) have a complete solution to bridge their real world presence with blockchains of Web3. This is what DePIN brings to the table, an umbrella term for technologies which decentralize control and ownership of physical infrastructure in the real world.

In summary, decentralization at a fundamental level has not occurred in Web3, consequently whatever is built on top is subject to the same fate as its shaky foundations.

2. **Web3 is not sufficiently powerful to independently run digital social networks, global-enterprise, and global-commerce**

   Web3 is missing necessary capabilities to run the digital world we live in. Namely big data, access to large arrays of compute (CPU & GPU), and secondary non-critical supplementary technologies such as AI (though recent steps° are getting closer). As Web2 is managing to a satisfactory degree and Web3 does not boast the computation capabilities of it's Web2 predecessor, there is not yet a convincing argument for migration.

   Web3 has something Web2 doesn't - smart contracts. These can offer Turing-complete compute of immutable code. However their power is limited and *they*, e.g. Ethereum Virtual Machine (EVM), are gas intensive if requested to process any enterprise-scale compute. This is not financially viable.

   Smart contract systems are also not homoiconic, they cannot change the overall system from inside. Old computers (circa 1960′s) used to be programmed by plug boards, an operator could not change the program from within the program, literal breadboards. That is to say the EVM is a plugboard, ZKEVM is also a plugboard, so is Bitcoin. They can't be changed from within themselves, for example all bitcoin nodes must update to account for changes. This poses technical problems for joining on-chain and off-chain worlds.

   These are more reasons to bring Web2-esque levels of compute power and composability capabilities into the on-chain world.

3. **Web2 public cloud's advantages outweigh those of Web3 P2P in its current form**

   Leading on from the previous point to expand on why there are few strong incentives to adopt Web3. History has shown humans will use convenient, cheaper, reputable and insured centralized systems at the cost of privacy. This holds until the systems break, by which time it's too late - 'Don't fix what isn't broken.' Though, things do break as was seen in the 2024 Cloudstrike incident°.

   Public cloud's flexibility (E.g. AWS EC2), reliability, efficiency, increased performance, time to production, and cost savings outweigh Web3′s currently achieved benefits. A decentralized cloud that can compete with public cloud is yet to emerge, the centralized control remains a choke point for the decentralization paradigm Web3 promises.

4. **Most widely-adopted applications rely on off-chain data**

   Businesses control data, data is a business. It is monetized and at the heart of many Web2 business models. Web3 promotes a need for 'free' data. Free in terms of monetary value and freedom of access.

   In the current set up, data providers are at liberty to rate limit data transfer, potentially price gouge with sufficient specialist-data monopoly, or turn off the tap entirely.

5. **People remain a problem**

   Major Web3 projects which either provision digital resources or control the distribution of technology behind oracles are run by companies, not the community or users. Most currently fail because they lack a trust mechanism. Trust is centralized on a single user or entity. Resource provisioning systems and oracle dependent networks could benefit from a reputation system in addition to decentralized governance.

   Decentralized Autonomous Organizations (DAOs) are a solution to leverage blockchains and smart contracts for management of governance, finances, and more for Web3 projects. They distribute decision-making, management, and ownership.

6. **Web2′s design will not scale**

   In the not too distant future our planet(s)[5] won't just have human agents and their associated machines to interconnect. Artificial Intelligence (AI) agents and the Internet of Things (IoT) will see this network increase exponentially to perhaps a quadrillion devices. We need an Internet that can scale for that quantity and demonstrate self-healing properties. The current topology of the Internet cannot cater for this expansion. The best design consistently evidenced in science and nature is a 'mesh' due to its scaling and self-healing properties.

[5]SpaceX - Mission to Mars and Beyond°

2. Moosh - The Design Blueprint

2.1. The Philosophy Matters

Beyond the overarching principle of decentralization and the above list of issues facing the industry two other factors contribute to the motivation underpinning the project.

Influencing the design of our solution are the facts that (a) adoption for Web3 is lacking due to complexity and lack of compatibility and (b) navigating cloud in any form is very difficult if the user is not technical.

Moosh considers how to achieve the overall mission for decentralization and the needs of the everyday cloud user with minimum compromise.

2.2. What Moosh Fixes

Moosh's proposal is twofold:

- A Reputation Network for Digital Social Systems & Commerce
- Real Decentralized Cloud Computing

Building a decentralized cloud remedies the centralization-weakened infrastructure pillar of Web3 entirely (Figure 1). DApps on Web3′s application layer can be hosted on the new cloud emergent from the aggregation of Moosh network's infrastructure providers. This same infrastructure can, in turn, help fix the tx layer of Web3 by decentralizing the operation of blockchain nodes.

It is anticipated that decentralized data centers and data services will come to the fore, also remedying the Data pillar of Web3.

2.2.1. A Reputation Network for Digital Social Systems & Commerce

The fundamental problem of trust is poorly solved by existing Web3 technologies, we target a P2P network for decentralized reputation tracking based on nodes' social and commercial connections.

Identities in the Moosh Network correlate to a brand, which is built up over time by commercial entities through forming connections with partners and customers.

The reputation of a given identity is calculated from a specific point of view, or perspective, based on the number of trusted connections and the social pathway through those connections to the identity. I.e. every node in the graph has their own opinion of it's surrounding connections.

The primary decentralized commercial application of this network is a P2P cloud network to facilitate an order book for digital resources.

2.2.2. Decentralized Cloud Computing

The Moosh Network will utilize blockchain technology to facilitate buy and sell orders for digital resources, such as a sell order for a virtual machine with a set resource specification.

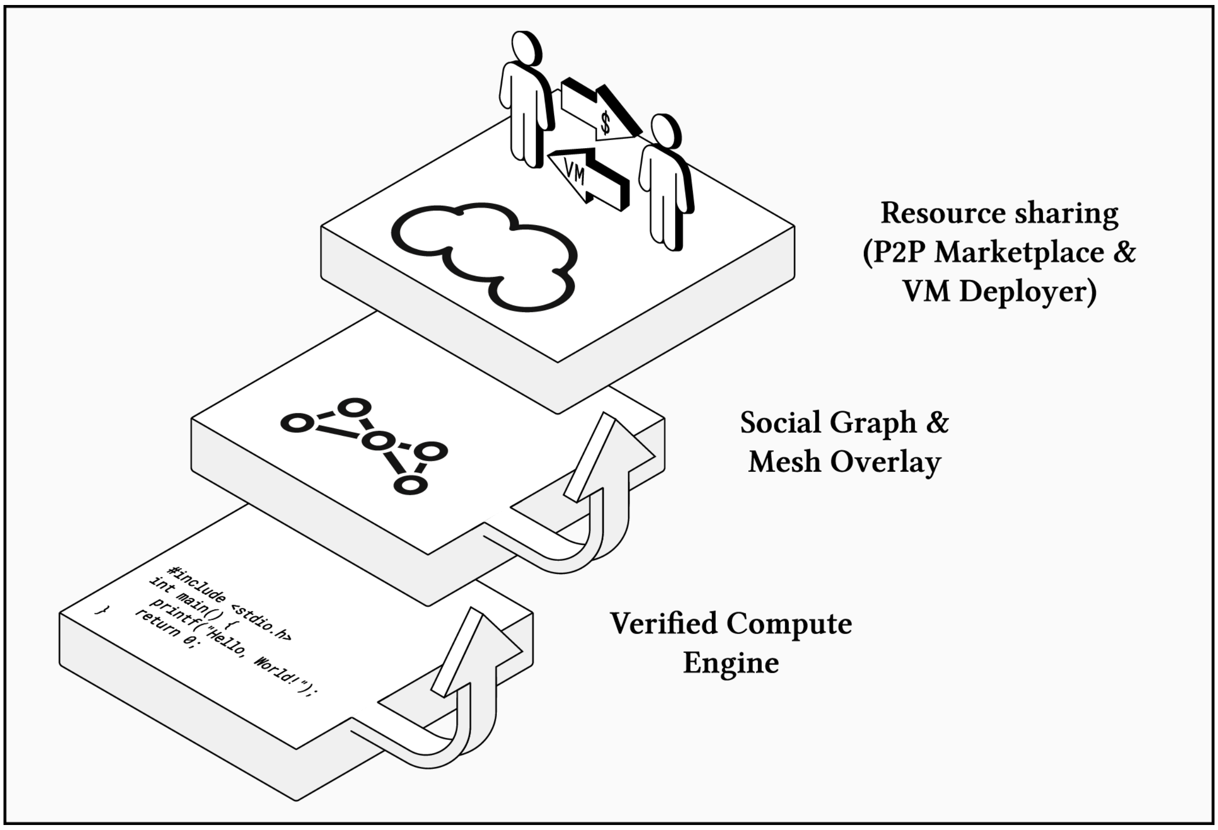
See later section Section 3.2.4 about Fully Programmable Provisioning (FPP). FPP combined with the marketplace is used to tie in not only personal hardware, but also public cloud providers and established Web2 data centers which have fiber optic connectivity for fast data transmission to quickly and seamlessly interconnect physical and virtual digital infrastructure. In turn this effort accelerates the viability of Web3 to serve global digital ecosystems and closes the gap between on-chain and off-chain compute limitations.

Given several limitations with the cloud industry at present and major issues with the way our Web3 industry handles our infrastructure, we believe the future of the cloud is decentralized. In any industry mistakes can happen, which is why it can be a good mitigation to spread certain infrastructure across multiple providers. In a decentralized cloud marketplace, use-cases such as redundancy can be baked in and frictionless, leading to a more robust digital infrastructure for everyone.

2.3. Building Moosh

The final version of Moosh has three layers:

1. P2P marketplace for digital resources
2. Social network (graph with trust scoring) + Mesh Overlay
3. A novel Verifiable Computation solution (VCE)


$
VM
Resource sharing (P2P Marketplace & VM Deployer)
Social Graph & Mesh Overlay
Verified Compute Engine

```
#include <stdio.h>
int main() {
  printf("Hello, World!");
  return 0;
}
```

Figure 3: The final version of Moosh Net has three layered components. 1. A client and provider facing P2P resource sharing marketplace. 2. Two open overlays: a trust-score laced social network and a mesh overlay network. 3. An engine chosen to best address the verifiable general computation problem.

An important point here is that layer 1 (marketplace) works independently, but is improved by layer 2 (open overlays). The third layer combines existing and experimental technologies improving both layer 1 and 2 tremendously.

The marketplace fosters growth of decentralized cloud - both Data and Infrastructure. Layers (2) and (3) from Figure 3 combine to support the marketplace but could be extended the target any on-chain to off-chain interface generically. All three layers will transition to DAO governance to solve for human-centralization issues.

In the marketplace, 'digital resources' range from general single items such as storage and compute (CPU, GPU), emergent services such as ZK-Proof-as-a-Service (ZKPaaS), through to full VMs. For example, the marketplace may later extend to delegate ZKPaaS work for oracles running on weaker networked VCE machines if it becomes economically advantageous.

2.3.1. Resource Sharing Marketplace

Moosh will implement a simple resource sharing marketplace with two classes of users:
1. **Clients**: Provision network resources and assign ratings to providers.
2. **Providers**: Lease out resources in exchange for payment.

Both clients and providers are participants in the social graph, they are the nodes in the system. The edges of the graph are the trust relationships, typically social connections, or commercial relationships.

Clients are the users of the network who pay to provision machines, they interact with the network through a desktop app where they can see available hosts and choose to provision one under terms defined in a smart contract.

Note we choose a desktop app because it gives Moosh important functionality like access to networking and local storage for the client to interact with their deployments more directly. In the desktop app each user runs the node daemon and stores their app's state locally which could include an address-book, system configurations, and past deployments.

Providers are users or entities which intend to provide cloud services to users over the Moosh network, they use the desktop app to view the status of their hosts.

Providers will create sell orders for their physical infrastructure or rented infrastructure using a server application which we initially intend on implementing in a virtual machine manager. Note that the virtual machine manager is separate from the desktop app.

If a client is directly aware of a provider (they're in their address book), it will automatically download it's available orders. From the list of known orders, the client will be able to directly issue a provision request.

Clients will be able to purchase a provider's sell order using cryptocurrency-enabled payments, which will give them access to a virtual machine on the host. If something goes wrong with this interaction they can then share disputes with the network affecting the provider's reputation with the client's social peers. This is also a disincentive for users to peer with 'anybody' or make untrusted connections, since they could have their provider list manipulated by illegitimate reviews and disputes.

As with existing cloud solutions, providers could offer shared memory or CPU resources to their clients at a lower rate essentially 'overbooking' their inventory, maximizing their resource utilization.

2.3.2. Social Graph & Overlay Mesh Network

Any human element of a P2P network requires trust between parties, particularly when there is an exchange of value. Even secure layer 1 cryptocurrencies rely on users trusting that the underlying wallet software is not malicious and that addresses aren't altered in transit.

In the case of cloud computing, where general computation can't be verified independently, trust is the only metric to judge cloud providers. Web2 cloud customers use Google Cloud Platform (GCP) or Amazon Web Services (AWS) because they trust that when they pay these providers they will get the advertised services.

In Web3, where contracts and legal agreements are harder to enforce, a lot more guarantees have to be made by third party providers for customers to rent software from them. To this end, Moosh introduces a peer to peer reputation system in our solution stack to explicitly lay out these trust relationships without relying on any single party. This allows anyone to share feedback or become a provider without a middleman.

In our system, clients generate reputation scores for known resource providers, this could be public cloud providers, individual server owners, data centers or similar entities. Similar to page rank in Google, this algorithm will use all available data to determine a provider's trustworthiness. This includes past deployment history, any stakes in the network, and crucially a weighted average of client's trusted peers.

Over time, this allows for a larger social graph to form where there are enough trusted providers that clients can provision servers with relative confidence that they will be reliable. There are in fact two constituent 'open overlays' in the Moosh network, consisting of (a) the social graph and (b) a mesh overlay network. See figure 3.

The social graph is defined by the user's manual connections as well as staked relations where the stake implies confidence in the provider by the client, similar to a customer's deposit before deploying cloud services. The social graph is traversed for discovery of infrastructure services and provides a means of attestation for commercial deals and product reviews.

The mesh overlay exists to provide the network with on and off ramps to the main Internet. Nodes which have a public IP address will host a virtual private network to relay messages between different subnets and privately accessible clouds. The overlay network is structured as a wireguard mesh network where peers with a public IP address can host an endpoint to route traffic between the public Internet and provide public access for peers that do not have a public IP address to run an endpoint. Having these entries and exits to the internet ensures that deployments in MooshNet can be made accessible over the public Internet for users to use their own domain names for their MooshNet host.

2.3.3. Verifiable Compute Engine (VCE)

Whilst verifiable compute is currently a research problem with active developments in the field, Moosh intends to utilize existing solutions and make our own contributions to the problem. The Verified Compute Engine and Open Overlays are co-operative and symbiotic, therefore any advances in the VCE layer of the solution stack permeate through to end users of applications and services operating on the surface. E.g. the marketplace.

More information about Moosh's experimentation in this field will follow in future publications.

3. Minimum Viable Product (MVP)

Initially, MooshNet's primary goal is to create an on-chain marketplace where sell orders offer a specification of digital resources such as, but not limited to, a NixOS° Virtual Machine (VM). This choice is to bootstrap immediate usefulness of the network to solidify the project's expansion and longevity. See section Section 3.2 for more details.

Once a sell order is offered to the market, it can be consumed by a user's deployment of software using agreed rules defined by each party, such as stake/deposit amount, dispute resolution method and allowed software.

3.1. Clients and Providers

In the MVP, the desktop app will focus on usage by the client to provision into whitelisted provider machines in a small-scale network. This desktop application will automatically record and share ratings to trusted peers in the network, which will be the beginning of our social graph.

A provider runs a program (See Section 3.1.1) that automatically manages a set of NixOS VMs with pre-configured settings. This program will coach the user through a wizard to set up their inventory of VMs based on the resources that the host machine has available.

This app covers the fundamental resource-sharing interactions, including capabilities for inventory discovery, provisioning, and configuration.

3.1.1. Virtual Machine Manager

The most basic implementation of a peer-to-peer (P2P) compute marketplace will use Nix to encode the virtual system being traded, to facilitate this we need software to run on the host machine and build the Nix code into the desired system. Since the Nix language provides native support for hosting virtual machines (VMs) and containers, we prefer to host this software on NixOS.

We believe the natural evolution of the VM Manager is into a git-based agent that tracks the source code for deployments, these deployments will initially be a system specification in Nix but eventually we aim to support application-layer hosting where a developer can deploy their react site or node.js backend directly onto the network. This type of system provides the convenience and cost-effectiveness of serverless compute and deployment in an open peer to peer cloud marketplace.

Moosh provides the application to the user, the provider provides infrastructure and the network facilitates an infrastructure marketplace.

3.1.2. Resource Provider Inventory

During setup or configuration, a resource provider creates a collection of sell orders as a set VM specifications that the host can support, which is known as it's inventory. Once the provider's inventory is exhausted, there will be the maximum workload of virtual machines running on that machine.

In the desktop app, clients hunting for VMs see an aggregated view of all provider inventories meeting their requirements, or greater specification if the price is appealing.

3.1.3. Nix to Manage VMs

Moosh begins by using existing reproducible trusted technologies, e.g. operating systems (OS) that are capable of delivering secure software supply chains. Such as, but not restricted to, NixOS.

Moosh is an advocate of promoting free and open source technologies. Nix offers declarative VM system configuration, providing us with a way to quickly handle provision requests on a host while interacting with the hypervisor directly through the Nix API rather than through an entire virtualization platform.

Using Nix, Moosh easily integrates a git-based deployment system and manages entire system configurations declaratively, signed by the customer's wallet. We believe this can help us to develop a commercial-grade deployment platform for a peer-to-peer market. A NixOS system configuration can specify software packages, systemd units, remote access credentials and in the virtualization module there is capability to allocate the GuestOS'[6] RAM and Storage.

We also envision a social aspect of being able to share a system configuration between individuals or businesses as reproducible 'blueprints'. A library of community-built blueprints will be continuously curated by the Moosh DAO in the desktop application.

3.2. Initial Use Cases - Bootstrapping Network Adoption

All Web3 networks suffer from a cold start problem, there's lots of networks out there, why is ours the one to adopt?

To address this, Moosh will focus on building the most immediately usable and beneficial part of the solution stack first. Following this solution, Moosh will build bridges to the existing cloud computing industry to provide compatibility and ease of migration. By creating standardized deployment methods, Moosh helps users to achieve redundancy and migration between providers.

In general there is monetary value in reducing counter-party risk, Moosh mitigates trust in third parties through collective bargaining. Using Moosh as a foundational fabric to build services creates immediate reputational consequences for typical concerns of cloud compute clients such as poor reliability, loss of data and changes to agreed terms.

3.2.1. Bridges

To support the economic viability of the network, we intend to facilitate bridging to traditional cloud providers through provider entities who can make existing cloud services available through the Moosh network.

Providers on the network can translate a MooshNet provision request to provider-specific API requests. Through this, MooshNet is able to host a superset of all of the existing cloud rental or VPS services.

[6]A guest machine refers to the VM on a host

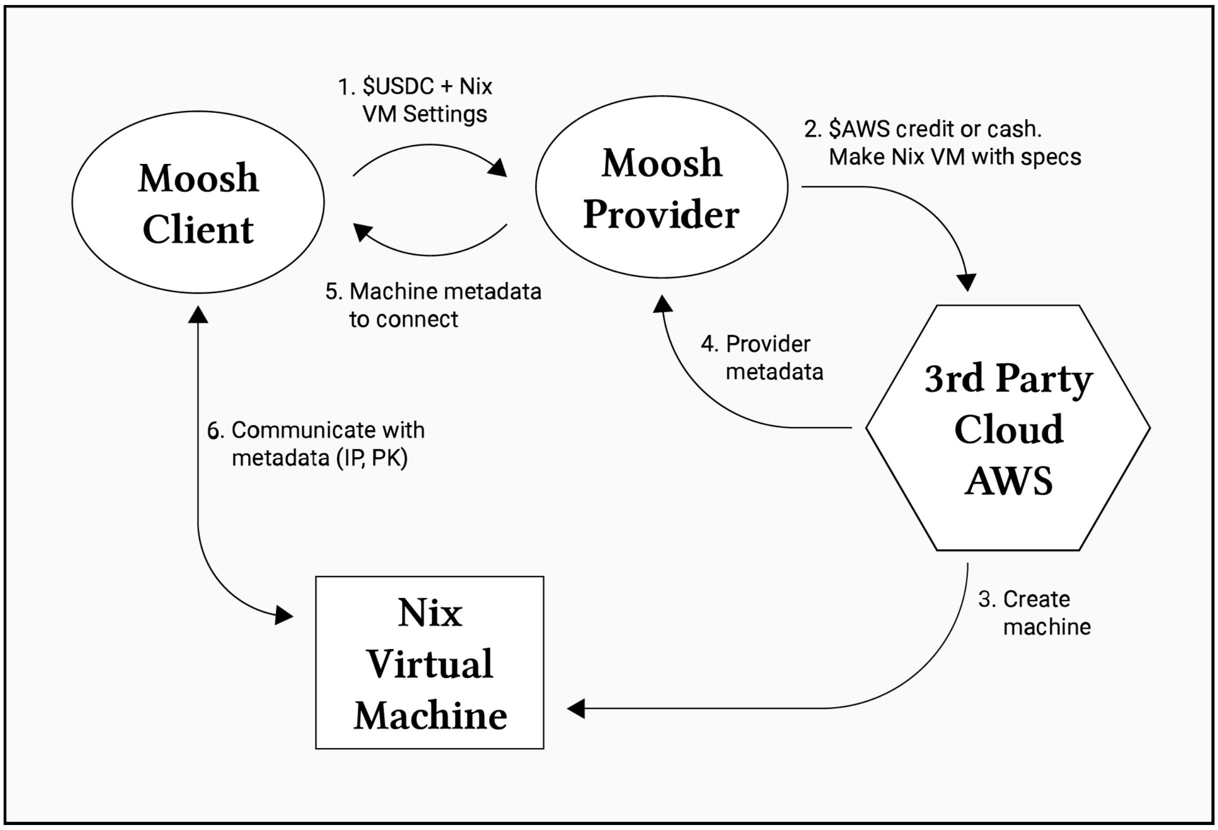

1. $USDC + Nix VM Settings
Moosh Client
Moosh Provider
2. $AWS credit or cash. Make Nix VM with specs
5. Machine metadata to connect
4. Provider metadata
3rd Party Cloud AWS
6. Communicate with metadata (IP, PK)
Nix Virtual Machine
3. Create machine

Figure 4: How bridges function.

As Figure 4 shows, a Moosh client provides a machine using the MooshNet protocol:

To start, the client issues a provision request to a provider on the network committing to pay crypto for a VM. The provider pays a 3rd party cloud (i.e. AWS) to actually provision the server.

Once the machine has been created and the NixOS VM has started, the cloud provider's metadata is shared to the Moosh provider, then given to the client.

With this information, the Moosh client is able to connect directly to their VPS (through SSH or other system) and start using the machine with their specified software.

Advantages of this system:
1. **Crypto payments**: Ability to pay for existing clouds with crypto payments or receive crypto for providing compute services.
2. **No KYC**: Clients in Moosh will not require KYC to use the application and don't need an account with the third party cloud to provision services for them.
3. **Integrate cloud services**: All the provider needs to do is write a thin wrapper around the 3rd party cloud APIs and fund an account to get started.
4. **Cloud provider agnostic**: The Moosh client doesn't need to know how the provider creates the virtual machine, it's only concern is that a valid VM is created at the end of the process.

For providers, they can resell VPS machines for crypto with very low cost of maintenance. For clients, they can get access to most existing cloud providers without needing to sign up for individual accounts or register a credit card.

With bridges, we can build a very compelling reselling market and dramatically increase the early compute power of the Moosh network.

3.2.2. Link-based Invite System

A link-based invite system, similar to magnet links or telegram invites could be developed to share providers or clients in the network.

These links are a hook to the desktop app to prompt the user with a connection offer. The prompt will include information on the offer and a confirmation button. After confirmation, the connection will automatically take place.

Should someone click an invitation URL without the app installed, they will be first prompted to download the app.

This small feature will enable a very powerful advertising use case, where providers can share their spare compute offers on various social media channels for users to discover them.

This also creates a low-cost on-boarding feedback loop where providers who want customers will push new users to download the desktop app through compelling offers.

3.2.3. Incentivizing Utilization of Spare Computers

Right now, if someone wants to make a new cloud provider, they have to:
1. Source the machines (inventory) and set up the virtualization.
2. Set up a website for management/admin, a payment portal, account management system for customers to rent machines.
3. Get users to trust the site through advertising, heavy discounts, and good-will.

This keeps a lot of otherwise viable compute out of the market. Potential small to medium size operations which can't afford to build a whole payment and management portal (2) or can't afford the marketing or time investment (3) are kept out of the market.

With Moosh, this is all streamlined to bring them in.

For (1), using the Virtual Machine Manager (Section 3.1.1), we simplify virtualization of the hardware.

Next, for (2) we completely do away with the need for a website at all, the provider will only need to be running the management program to announce sell orders in the system and manage their funds through Web3 wallets.

Finally, for issue (3) Moosh simplifies management of prices, and publically shows trust relationships. The provider would initially have to advertise their offers on the internet using the link sharing system (See Section 3.2.2) which will even onboard people who don't yet have Moosh installed. After building up a good reputation, the provider will be able to increase prices as the good-will has been built.

Having addressed the biggest concerns in making a new competitive cloud provider, Moosh will enable small to medium size cloud operations to join the market. This will increase competition and lower prices for everyone while decentralizing compute at the same time.

3.2.4. Fully Programmable Provisioning (FPP)

As mentioned in Section 1.5, Web3 adoption is lacking due to technical complexity and a lack of compatibility. These traits also apply to traditional cloud's resource provisioning. If a user is not technical, it's very difficult mixing and matching resource providers which is further complicated between competing walled gardens.

With a reliable P2P protocol, all provisioning activity and cost analysis can be managed automatically. This to an end user means automated programs which select for the best/cheapest/most-reliable compute on demand from a sprawling marketplace of providers you trust.

Firstly, having a reliable P2P protocol removes vendor lock-in. Lock-in is when vendors intentionally make their software incompatible with one another to trap users in their ecosystem. This keeps users from leaving and makes cloud providers hard to compare, as their offerings aren't exactly identical the price tags don't mean the same things.

Moosh enforces a common standard (The protocol) that providers must support, so they can only compete on reliability, features, and price. The user is given a free UI which works across providers, so they can always choose to migrate elsewhere and aren't trapped in one cloud ecosystem. This greatly reduces vendor lock-in and increases competition across providers.

Another benefit of this system is full programmability across all providers. This means you can write a program of arbitrary complexity to provision or manage servers across the whole network. For instance, provisioning machines only on certain dates or when finances are at a certain range or to poach machines that are discounted. It's up to the user to decide.

What makes this even better is the use of cryptocurrency for payment, which means even payments can be completely tuned to the users liking. Clients aren't limited to using a checking account, they can have a fully dynamic system where their spending is partially limited or subsidized by smart contracts.

Having a public, consistent protocol with crypto payments both increases competition and makes cloud provisioning more dynamic and customizable.

3.2.5. Potential Future Use-cases

With Moosh, an individual resource provider could provide the full convenience of commercial public cloud solutions using NixOS to provide the user purely with the 'application layer' similar to existing serverless solutions, but implemented as a decentralized solution with the benefits of direct market competition for running those functions.

We could see decentralized data centers or data lakes emerging in our network as the Moosh equivalent of mining pools. The reputation system enables this emergent practice as linking provider reputations can combine their customer reach for mutual benefit.

3.3. Tokenomics

After initial MVP development, Moosh tranforms to a DAO governed DePIN project. Full tokenomics information will follow in a later publication.

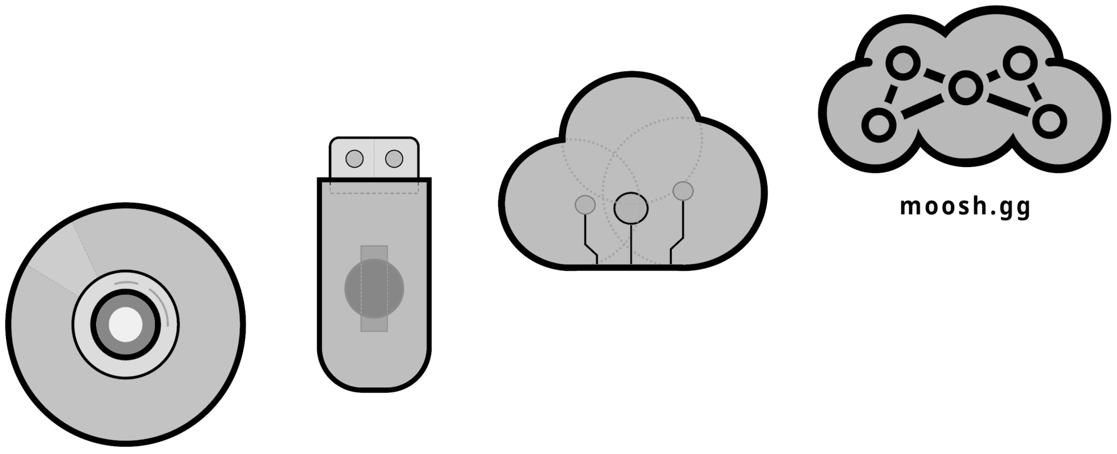
Tokens in the Moosh ecosystem serve cloud utility and in the future will extend to the verified compute engine. For the marketplace, providers stake to compliment their trust scoring and are assessed by the DAO on a transaction basis. The DAO can slash a provider based on malicious activity or substandard performance. Globally adjusted staking coefficients are automatically calculated per emission epoch and frequently monitored by the DAO to avoid social engineering attacks.

This coefficient is factored into the reputation score calculated by the clients, so it serves as a form of decentralized provider advertising across the whole network. They can decide how much they factor in the stake coefficient, which keeps the DAO accountable as the tokens utility is tied to the value of advertising on the network.

The DAO is interlaced with the trust tracking social graph via slashing mechanisms to discourage malicious behavior and incentivize high degrees of trustworthiness.

A decentralized talent marketplace embedded within the DAO is planned to track core and supplementary contributions with stochastic and value-oriented mechanisms. These will measure human task-based effort to partner with the social graph trust scoring. Human work and digital work are both tracked and rewarded in the Moosh ecosystem.

A road to long-term decentralization is always at the heart of the project's evolution to ensure longevity of the project.



4. Contact Information

Website : moosh.gg
X.com : @mooshnet
Email : admin@moosh.gg

moosh.gg

5. Acknowledgements

Cover artwork & Illustration: Mome
Illustration: Kalvin Cheung

moosh.gg