Stratosphere: A Distributed Cloud Network for Decentralized Application Scalability

Ryan Berkun, Mark Meras, Christopher Lee

Fabrx Labs

DRAFT Version 1.0.2

Abstract

Due to the limitations in cloud resources, modern applications are constrained on the current blockchain technology stack. Per-application database and storage supported on Ethereum is in the order of kilobytes. Meanwhile, maximum per-application compute supported on Ethereum is in the order of 0.00075 calls per second (based on a total network compute of 15 smart contract calls per second, with 20,000 active smart contracts to-date). As the number of active smart contracts on the Ethereum network increases, the maximum per-application calls per second inversely decreases.

This paper proposes the Stratosphere Network, a distributed cloud network. In the network, cloud operations are performed by selected nodes, in which each node runs a replication of a given computer function. This function can be any cloud operation, including database, storage, and compute. With direct integrations of each node into a major cloud provider, the distributed network creates a "trustless layer above the cloud".

Now, decentralized applications launched on the Stratosphere PoS blockchain are equipped with per-application database and storage up to 16,000,000,000x greater than that of Ethereum to-date and per-application compute up to 1,300,000x greater than that on Ethereum to-date. Furthermore, the financial cost savings of Stratosphere, as compared to Ethereum, are significant - up to 8,695,652x less for storage, 1,739,130x less for database, and 150,000x less for computation.

CONTENTS

1. Introduction

This paper introduces a design for a distributed cloud network, built on top of a new primitive called **Trusted Functions**. Trusted Functions are an extension of consensus validation, where node output is utilized to verify the proper execution of cloud services.

Trusted Functions serve as the base for all further Proof of Cloud (POC) consensus validations. This includes Proof of Compute, Proof of Database, and Proof of Storage. The distributed network of cloud service nodes and consensus validation forms a trustless layer atop current cloud resources: the **Stratosphere Network.**

We outline three general application categories where trustless cloud resources provide immediate and immense value: Blockchain Native Networks, Peer-to-Peer Networks, Multiparty Networks.

1.1 Reinventing the Cloud

Current blockchains attempt to reinvent the entire cloud service technology stack. This task, while idealized for decentralization, is infeasible due to the diametric opposition of decentralization characteristics and resource capacity [19].

2019 dapp survey [21] cited **48% of dapps rely on centralized cloud providers to run backend code** (computation), while **75% of dapps rely on centralized cloud providers to run frontend code**. In addition, other centralized resources, such as databases, are more popular than decentralized options (31% centralized database vs 25% decentralized database).

These statistics are **prior to scale** (eg. greater than 1 tx / sec), in which **33% of dapps have no plan**, **39% of dapps plan to use existing solutions**, including Layer 2 solutions which reside on cloud providers, and **27% of dapps planned to self build scalability resources**.

The node market further outlines developer attachment to centralized cloud providers, as **more than 60% of Ethereum nodes run on the cloud** [26].

The current state, where blockchain applications utilize a significant degree of centralized services [21, 20], exemplifies the **benefits of cloud computing today - inexpensive, fast, battle tested, secured by major brands**. Blockchain applications are willing to make sacrifices in decentralization in return for operational efficiency and ease-of-use [21].

The issue, for both applications integrated with blockchain solutions and those without, is how current cloud solutions act as a black box. For developers, trust relies on a single service provider, running an entire operation. This **"all eggs in one basket" approach is prone to systemic risks, including human error, malicious actors, and platform migrations**. **On the outside, platform partners, companies, and users are left in the dark** on events occurring within the application tech stack.

2. Ethereum, and Beyond

The Ethereum whitepaper outlined three categories of applications; financial, semi-financial, and non-financial [2].

Financial applications include token systems, derivatives, stable-value currencies, savings wallets, crop insurance, and smart multi-signature escrow. The explosive growth of the Decentralized Finance (DeFi) ecosystem in Ethereum has proved the function of Ethereum as a financial layer.

While semi- and non- financial applications have emerged, they are limited in scope. Only applications that function on minimal computation inputs are successful in these categories [3]. This is significant, **as over 87% of dapps are built on Ethereum** [21].

When compared to expensive options on AWS, **Etherum, at most optimal conditions, has 2,200x slower write speeds, 14,000x write costs, 20,000,000x more expensive storage costs, 903x slower bandwidth**. Due to these limitations of network scalability, Ethereum based non-financial decentralized applications with a high-engagement run into issues of high costs and high latency [19].

2.1 Cloud Resources Needed for Scale

Operational resources **at magnitudes 1,000,000x - 1,000,000,000x+ greater than those available today on public blockchains** are critical in achieving decentralization at scale. For example, **Uber**, at 5 years old, **held several terabytes** of data [1]. To compare, the entire size of the **Ethereum blockchain to-date is 450 GB**.

The **Stratosphere Network**, as built atop major cloud providers, will empower modern applications with immediately accessible, decentralized **resources at orders of magnitude of up to 16,000,000,000x that of Ethereum**.

3. Trusted Functions

To achieve a trustless, scalable network for high engagement applications, we propose our core primitive: Trusted Functions.

These functions act as a query on node operations and utilize a commit-reveal scheme to validate consensus on node outputs.

The Trusted Function is formed from three building blocks: a distributed cloud, a commit-reveal scheme, and consensus validation.

3.1 Distributed Cloud

A distributed cloud network is a collection of nodes, each run an identical version of an application-specific cloud service.

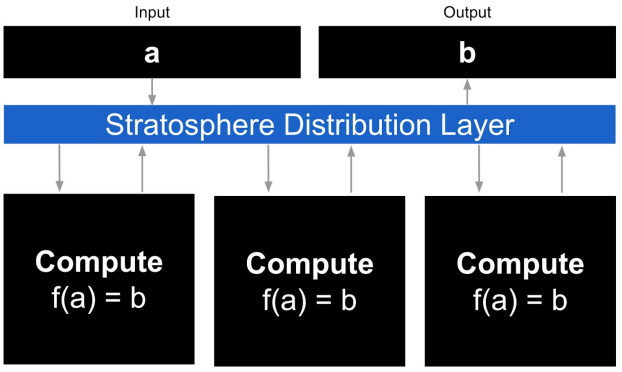

Figure 1: Distributed cloud network.

Figure 1 outlines the network architecture of a distributed cloud network for a compute service. An identical replication of a function, in which f(a) = b, is running on each node.

Note the addition of a distribution layer, the coordination framework to receive inputs, call application-specific nodes, and return node outputs. This layer resides in the node architecture and is accessed through the Stratosphere blockchain's JSON RPC.

Furthermore, the distribution layer removes the self-hosting of applications by developers. The Stratosphere Network propagates all application-specific cloud service interactions to the application-specific nodes. Thus, trust becomes distributed amongst nodes in the cloud network.

3.2 Commit-Reveal Scheme

The commit-reveal scheme allows all nodes to prove the performance of an operation. An immutable record of all node commit-reveal operations is maintained on-chain.

In this model, each node is obligated to run an application-specific query, hash the operation output, and commit a composite hash on-chain. A unique salt, as well as the application-id, is sent to each node from the query. The application-id, salt,

and output hash are hashed to create the composite hash.

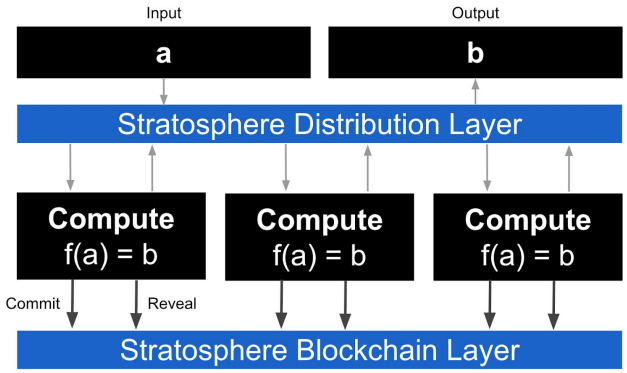

Figure 2: Commit-Reveal on Stratosphere Blockchain layer.

Figure 2 extends the depiction of a distributed cloud network to include a commit-reveal scheme.

On initiation of a commit-reveal scheme, the application generates X (equal to the node quantity) number of random salts. Each salt is hashed, with the key:value pair of node ID:app salt hash written on-chain. The application then sends the associated salts to each recipient node.

Step 1: Application Generates Salt Hash

```
[nodeID1, h(S)a1]
…
[nodeIDn, h(S)an]
```

Formula 1: Array of the key value pair of node ID and hash of application salt. The array is written on-chain. Where h = hash, S = salt, a = application, n = ID of the node.

Step 2: Application-Generated Salt Sent to Node

```
[Sa1 → nodeID1]
…
[San → nodeIDn]
```

Formula 2: Send of randomly generated salt from application to node. Where S = salt, a = application, n = ID of the node.

In addition, each node generates a random salt. Each salt is hashed, with the key:value pair of node ID:node salt hash written on-chain.

Step 3: Node Generates Salt Hash

```
[nodeID1, h(S)N1]
…
[nodeIDn, h(S)Nn]
```

Formula 3: Array of the key value pair of node ID and hash of node salt. The array is written on-chain. Where h = hash, S = salt, N = node, n = ID of the node.

Step 4: Node Calculates Composite Hash

```
IF computed  h(S)an ==  h(S)an on chain:
      Node with ID n calculates:
            output(n) = function(input)
            hn = hash(output(n))
            commit(n) = hash(nodeIDn + San +
                  SNn + hn)
      Then, commit(n) written on-chain.
ELSE:
      Application presumed to cheat:
            Node voluntarily doesn't
participate
            Node complains to reputation
system
            Node stops serving application
```

Formula 4: The node commit of calculated outputs. In place are catch guards to check if the application is cheating. Where h = hash, S = salt, N = node, a = application, n = ID of the node.

The commit contains the formula to generate the composite hash. If the salt is correct, the node will commit the composite hash on-chain.

Step 5: Application Processes Node Commits

```
IF t1 - t0 > commit timeout || all nodes
committed:
       Continue commit-reveal with committed
       nodes only.
```

Formula 5: Application waits for commit timeout or all nodes to commit. Where t0 = start time of commit, t1 = current time, commit timeout = predetermined timeout of commit by application, n = ID of the node.

Nodes have a limited quantity of time to run the commit function. Only nodes who complete commit will be included in consensus.

Step 6: Application Writes Salts On-Chain

```
Sa1...San → Stratosphere Blockchain
```

Formula 6: Application salts are revealed on-chain. Where S = salt, a = application, n = ID of the node.

After all application salts are written on-chain, a reveal is called.

Step 7: Reveal: Nodes Write Variables On-Chain

```
IF:
          [Node ID, h(S)an, h(n), h(S)Nn] →
            Stratosphere Blockchain
ELSE:

Reveal call times out, node is not included in
             consensus payouts.
```

Formula 7: Reveal is called, obligating nodes to write all variables on-chain. If a node does not commit in the stated period of time, the node is not included in payouts and may be penalized. Where n = ID of the node, output = cloud compute result.

Reveal obligates each node to write on-chain their node ID, the hash of the application salt, the hash of the function output, and the node generated salt. All on-chain commits are public and immutable.

Step 8: Node Sends Function Output to Application

```
output(n) → Application
```

Formula 8: Nodes send the output of their function to the application. Where n = ID of the node.

Step 9: Application Verifies Node Output

```
IF on-chain commit(n) != generated commit(n):
       Application complains to decrease Node
       (n) reputation score.
```

Formula 9: Application verifies node outputs based on on-chain values provided by nodes. Where n = ID of the node.

3.3 Consensus Validation

Consensus on hashed outputs is performed by a network of validators, the number of which is determined by the developer.

Step 10: Validator Consensus

```
IF output(n) of > ⅔ Nodes is equivalent:
       consensus = TRUE
ELSE:
       consensus = FALSE
```

Formula 10: If a super-majority of node output is produced, validators state consensus as TRUE. Where n = ID of the node.

The consensus variable for an application-specific commit-reveal is an on-chain boolean. If a super-majority of hashed outputs occurs, validators vote to set the consensus boolean to TRUE. If a super-majority is not reached, validators vote to set the consensus boolean to FALSE.

Step 11: Validators Slash Non-Performing Nodes

```
IF Node(n) is not in > ⅔ group:
      Node(n) does not receive payment.

IF Node(n) commit != reveal:
      Slash Node(n) stake
```

Formula 11: Validators slash the stake of nodes that do not commit and reveal the same answers. Nodes not in the super-majority do not receive payment. Where n = ID of the node.

Step 12: Validators Process On-Chain Updates

1. Validator selected to propose block
2. All validators run on-chain commit-reveal to verify transactions
3. Validators vote to add the node payments and global reputation score updates onto a new block.
4. If supermajority, then validators pass new payments and reputation updates. Else, validator recourse ensures and node payment is locked in escrow until the resolved decision.

Formula 12: Validators process the results of consensus, initiating node payout and updating the global reputation score.

3.4 Trusted Function

The combination of all three building blocks enables the execution of a Trusted Function.

Initially, the developer will set the frequency of (randomized) Trusted Function executions. Subsequently, Trust Functions can be called by any member on the Stratosphere Network, provided proper payment is supplied to the validators.

As Trusted Functions necessitate on-chain execution, the historical record of node operations and the consensus is permanently available for verification.

A track record of inclusion, or exclusion, from the super-majority of TRUE consensus operations, will develop for each node. This track record acts as a signal of trust.

3.5 Prevention Against Attack Vectors

Applications and nodes have two main drivers that lead to improper acting: maximize monetary benefit and targeted malicious acts.

Monetarily, app developers may attempt to minimize spend with maximum output, while nodes may attempt to maximize gain with minimum work.

Both app developers or nodes may perform targeted, malicious acts, even when not monetarily profitable.

Thus, we detail prevention mechanisms to combat attack vectors and maintain trust.

3.5.1 Output Pre-Process by App Developer to Halt Trusted Function Without Node Payment

Application developers are financially incentivized to compute the output of each node, prior to node payment. The output of a hash can be easily computed if a Trusted Function is run on a binary input.

We introduce the node generated salt, the hash of which is committed on-chain, to prevent application developers from computing the output answer prior to node payment.

3.5.2 App Developer Falsifies Sent Salts

In an effort to maliciously target nodes, application developers may send false or no salt to specific nodes.

To disincentivize malicious acting, we introduce the Application rating system. Nodes can submit a rating on the application to the validators prior to consensus computation.

In addition, nodes may simply drop the application if the application developer is continuously targeting them individually.

3.5.3 Node Computes Output from Other Nodes

As the committed hash is known on-chain, other nodes may use public information to compute the output. This would remove the necessity of the node from truly performing the cloud service.

Thus, a unique, application generated salt is sent to each node. Assuming no collusion amongst nodes, the hashed function output of each node is computationally infeasible to discover.

3.5.4 Node Stall of Hash Output Reveal

If a node is not performing the cloud operation, the machine is incentivized to stall the network reveal.

To counteract, we introduce the reveal timeout, in which each node must reveal their hashed output on-chain within a specific timeframe. Nodes who do not reveal are penalized either by lack of payment or slashing.

3.5.5 Node Composite Hash Does Not Compute into Node Operation Hash

Nodes who commit a composite hash that does not compute with their operation hash are penalized by lack of payment or slashing.

4. Stratosphere, A Distributed Cloud Network

Stratosphere utilizes Trusted Functions to create a distributed layer of transparency atop current cloud providers.

Nodes within the network may support any cloud provider available through the network's JSON RPC. Initially, the network plans to support Amazon AWS [4], Microsoft Azure [5], and Google Cloud Platform (GCP) [6]. Adaptors for additional cloud platforms can be created by third-party developers for use in the network.

The hybrid nature of Stratosphere enables developers to build public, secure, decentralized, scalable, and trustless applications atop familiar infrastructure.

4.1 Stratosphere Network

We propose three initial cloud services to be supported on the Stratosphere Network: compute, database, and storage.

To an application developer, the benefit from the network is derived from the availability of necessary cloud services and the number of nodes available to host such operations.

4.1.1 Proof of Cloud

Through the Trusted Function, any cloud service can be substituted as the validated function. Proof of Cloud is considered valid when the node output is verified through consensus by the validator network.

While each cloud service may necessitate a specific query to perform a Trusted Function, the network architecture remains the same.

Nodes offer cloud services through the JSON RPC. Thus, creating a marketplace of providers being seamlessly made available to developers.

During set up, developers pay for the new service launched by the node and lock-in a pre-payment into an escrow contract. The service is then initiated across the number of nodes specified, with a publicly available application ID endpoint and a wallet owner address. This recognition system is similar to smart contract identification on Ethereum.

4.1.2 Proof of Compute

Compute resources are aimed to perform cheap, fast, and secure operations.

Proof of Compute enables a far broader computation set than current smart contract blockchains. Any application operation which functions properly in a lambda script or EC2 instance is initially supported.

Note that **developers are free to code in the language and program best fit** for their application.

4.1.3 Smart Contracts

Smart contracts on the Stratosphere network can be run as lambda functions, distributed amongst a network of nodes. This vastly enhances the computation flexibility of decentralized applications. Specific smart contract types include Sprint, Compute, Logic, and Conditional.

Sprint Contracts prioritize speed. For operations that need to be executed in a timely manner, lambda functions can be written to minimize processing time.

Compute Contracts are built for operation intensive functions. For operations that handle outside API requests, data processing, or complex loops, lambda functions can be used to pull together all necessary resources.

Logic Contracts are built for standard business logic. For operations that process multiple layers of business logic flows, lambda functions can be written as light operations to direct data.

Conditional Contracts are simple "If Then" statements that can be run quickly and independently. These lambda scripts enable builders to quickly connect "If This, Then That" functionality into their application. Conditional triggers can include in-application, on-chain, off-chain, and third party events.

4.1.4 Smart Computes

Standard computation can be run through distributed EC2 instances. Docker containers are the ideal runtime container for these processes as they enable true flexibility in software development. Necessary computation that is too heavy for the Ethereum blockchain, such as processing an off-chain orderbook, can be seamlessly run in a distributed, trustless node network.

Large computation processing can be utilized to fulfill specific needs. The breadth of services spans from general Digital Ocean droplets, running a distributed network of Ubuntu machines, to the Amazon SageMaker, for machine learning computation.

The ability to decentralize large computational resources enables powerful trustless applications in the areas of AI / ML, Gaming, VR / AR, and biological/mechanical processing.

4.1.5 Proof of Database

Proof of Database services are aimed to mirror the use of current database infrastructure. Initial planned database types are PostgreSQL and DynamoDB.

The database is synced across each node at the same rate. Each node maintains a list of all database commands and the current database state.

New database commands, i.e., new transactions, are sent to the network's JSON RPC. Each database command is distributed to all nodes running the application-specific database. The command itself is added to the list of all database commands, as performed on each node's specific database.

4.1.6 Proof of Storage

Proof of Storage on the Stratosphere network will support both centralized and decentralized service providers. Initially, the network will support S3, Google Storage, Azure Storage, and IPFS.

The distribution of stored files across multiple storage networks increases an application's decentralization and censorship resistance.

In addition, developers can decisively select between centralized and decentralized storage services for the various parts of an application.

4.2 Stratosphere Blockchain

The Stratosphere Blockchain is necessary to maintain an immutable public ledger of Trust Function processing and validation.

Each cloud service has a randomized frequency, established by the developer, at which Trust Functions are performed.

In addition, the network's native token, stablecoin, token-related smart contracts, and user wallets all reside on the Stratosphere blockchain. Tokens, wallets, and smart contracts can be referenced and engaged with through application layers of the network.

Stratosphere operates as a Proof of Stake blockchain. We utilize Tendermint [7], where a known validator is selected to propose the next block. Voting on acceptance of the proposed block works through a super-majority in which > ⅔ of validators must reach consensus on a new block.

Delegators can stake to a validator to participate in staking rewards and contribute delegated security to the network. Total delegated stake influences validator vote weight, which is a function of the bonded token to the validator, divided by the total token bonded to all validators. The slashing penalty for untruthful behavior is imposed on both the validator and the delegators.

Due to the inherent increase in computation complexity correlated with validator quantity, a validator number limit will be imposed. Resulting in only validators with a stake greater than the smallest validator in the set to be eligible to earn rewards.

As previously stated, the Stratosphere blockchain will be built on Tendermint, which provides a secure, Byzantine Fault Tolerance consensus engine, built to scale Proof-of-Stake blockchain. Tendermint enables high performance, handling up to 1,000 tps, and instant finality, as transactions are finalized on block production.

In addition, launching with the Cosmos Network [8] warrants the Stratosphere blockchain instant modularity, EVM connectivity, and cross-chain interoperability.

4.3 Members

Nodes, validators, and developers interact to create the distributed cloud network of the Stratosphere blockchain.

Nodes create the operational layer where distributed cloud services are performed.

Validators bridge the operational outputs into a consensus layer on the Stratosphere blockchain.

Developers utilize cloud resources available by the operational layer to create decentralized applications.

4.3.1 Nodes

The overall function of Stratosphere nodes is twofold:

1. Perform cloud service operations that ideally offer a breadth of services across an array of providers.

2. Secure the network through distributed computation amongst many nodes.

Greater node distribution offers increased flexibility and security for developers, companies, and end-users.

To supply cloud services, each node must connect itself to a cloud service provider while simultaneously connecting a payment mechanism to the cloud service account.

This creates an innate KYC filtering (as the node must sign up for the cloud service platform) that provides greater security and brand authority to the network.

To counterbalance this partially permissioned layer, Stratosphere is cloud provider agnostic allowing open-source developers of all types to integrate other, less permissioned, service platforms into the Stratosphere Network.

At the cloud service layer, Stratosphere provides service adaptors for nodes to securely connect a cloud service to the network. In doing so, the cloud service offered by the node is automatically available for consumption by any member of the network.

Fee collection for nodes occurs at the blockchain level of the Stratosphere network. The process works similar to the payment mechanism of cloud services today, pay-per-use. However, in the Stratosphere network, cloud service resources are pre-paid.

On pre-payment, both the native token and stablecoin are sent to an application-specific escrow contract. The total payment amount needed is calculated by the summation of all fees charged at standardized timeframes or requests by application chosen nodes, multiplied by the number of timeframes or requests.

Node payment occurs during Trusted Functions execution. Prior to the main Trusted Function, a Billing Trusted Function is run to query each node on its specific cloud usage. The amount is hashed on-chain and used in the payment delegation by validators.

4.3.2 Validators

Validators serve as the verification bridge between the application layer and the blockchain layer. Validators perform two major functions: (1) verify consensus amongst nodes on application-specific services during Trusted Functions execution and (2) vote on the addition of new blocks to the chain.

The role of validators at the application layer is unique to the Stratosphere network. While the service performed is constructed in a similar fashion to the main consensus model, the reward mechanism is driven by prepayment from developers.

During the execution of Trusted Functions, validators state consensus, send transactions for node payment, update the global reputation system, and slash nodes based on on-chain salt + hash operations.

Building on research proposed by Oasis Labs in Ekiden [18], application-specific consensus is separated from the computational layer. Only a portion of the total network validators perform each application-specific consensus, enabling network scale.

The degree of Trusted Function frequency, thus the validator activity, in the application layer is set by the developer. A greater frequency of validation equates to greater trust levels assumed by the application.

On the main Stratosphere blockchain, validators work to secure the network, verify application hashes, commit new blocks to the chain and receive network rewards in exchange for their work.

4.3.3 Developers

The development environment on the Stratosphere network is aimed to be similar to that of Ethereum.

Node operations are callable through a JSON RPC interface. Stratosphere extends the node environment to launching application-specific cloud services.

In order to launch a service, the native token must be used to cover an initiation fee and staked in advance of usage.

Once a cloud service is launched, each node service has a unique service ID that is connected to a global application ID. A system similar to smart contract identification will be built to support the multitude of unique resource identifications.

To simplify the lives of developers, graphical user interfaces can be built on top of the Stratosphere network. The first interactive block explorer will be either released or commissioned by the Stratosphere Foundation.

Similar to Ethereum, the client design of the Stratosphere network is to serve as a Javascript API object. Consequently, user interface platforms are able to connect to the cloud resources of any decentralized application, directly from the client-side.

As described prior, usage of cloud services is paid in advance.

In addition, developers must also set the degree of decentralization they enable for their application. These settings include: (a) number of nodes per cloud service, (b) frequency of consensus hashed onto Stratosphere chain by validators, (c) allowed degrees of deviation from operation output vs. complete consensus needed, (d) enforcing native token fee paid on platform usage (not required, but can be used to monetize protocol and limit DDOS attack risk).

The downside to greater decentralization is higher costs on development resources, as multiple nodes are paid to run cloud services. However, as outlined in the *Financial Cost Savings* section, we project this cost to be 1,500x - 8,000,000x less than that of other blockchain alternatives.

4.4 Free Market System

We model the marketplace bidding framework on that proposed by Akash Network [22], and extend upon it by introducing an operation and provider-specific deployment order system. Developers propose the launch of a specific service, on a specific provider, in a behavior similar to deploying on a current cloud platform.

In the free market system, nodes must compete to be chosen as providers. Per cloud service, developers select the maximum number of nodes, total funding, and time-frequency of Trusted Functions (which equate to node payouts).

While nodes will eventually be able to integrate with a heterogeneous array of major cloud providers, the network will initially support a single cloud provider per operation. The node rate per cloud service is to be set by the network. For example, AWS lambda will be set by the network at $0.000033334 / GB-Sec (double the cost of a standard lambda call).

The free market system necessitates developers to pre-pay for application-specific cloud services. Necessary payments are divided into three types: initiation, usage, validation.

Initiation Payments are the amount paid by the developer for a node to launch the cloud application. These initial payments create a mechanism similar to ETH gas fees and are aimed to minimize repetitive, low yield usage and prevent DDoS-esque attacks on the network.

Usage Payments are the amount paid to a node per payout frequency. Payout amount is set by the nodes. Node costs are paid in the native stablecoin, while node profit is paid in the native token.

External Request Payouts are a minimum base payout set per external server request. This payout is to prevent the developer from performing a malicious DDOS attack through the node. In addition, nodes will have an internal wrapper for these calls with specific limits on DDOS signaling parameters.

Validation Payments are the amount paid to a validator per consensus validation run on a Trusted Function. This consensus operation writes the result hash on the Stratosphere blockchain and is the unique, second role of validators in the Stratosphere network.

Total validation payment per consensus validation is pre-funded by the developer. Validators offer to perform for each round with the validator reward equating to the total pre-fund divided by the number of validators. A validation payment floor amount can be established to ensure an incentive for validators to perform.

If a greater number of validators subscribe to an operation than the total payment available, a randomized selection will designate the validators for this specific round.

4.5 Reputation System

Brand based node reputation systems are proposed to strongly incentivize proper behavior and performance [23]. We attribute reputation scoring metrics to both the Application ID and Node ID of the Stratosphere Network.

Nodes can be identified by the smart contract used to receive payouts, labeled as node0x… . As nodes must both register and pay for the cloud providers supported, the ability to cheaply spawn psudoanonomous node identities is naturally limited. Node metrics include minority reporting [%], uptime, staked STR amount, # applications hosted, and ID's of hosted applications.

Applications can be identified by the smart contract used as escrow for node payouts, labeled as app0x… . Since applications can be cheaply spawned, they must build reputation through various network methods, including open-sourcing code, staking, uptime, active use, flag types, and # outbound requests.

The reputation system provides an adequate solution to issues both on blockchain attack vectors [13] and cloud attack vectors [14]. Sybil attack is mitigated when node reputation is built over time [15]. In addition, the ability of one node to launch an eclipse attack is diminished, as the time and resources needed to create a positive node reputation is non-trivial.

An application may maliciously use the node network to pursue cybercriminal activity. In this manner, each node would be at fault from the cloud provider for running such operations. To prevent such occurrence, the Stratosphere Network introduces a flagged application activity list that includes banned cloud operations, such as DDoS, phishing, and man-in-the-middle attacks. This list is continuously maintained and updated by the Flagged Application List Committee, as described in the *Governance* section below. By open sourcing code through the network, applications are able to

attain a higher reputation and enable members to verify the lack of cybercriminal usage. While malicious applications would accept lower reputation in lieu of open sourcing code, nodes do have access to the source code and are not restricted from publishing this code. We employ an insurance pool as (1) an incentive for both nodes and community members to prove cybercriminal activities of an application and (2) a potential reimbursement vehicle for damages incurred by nodes.

It is also recognized that many cyber criminal attacks are processed through outbound requests [16]. We propose a limited outbound usage based on application reputation, initializing at 10 requests per minute [17]. For uncapped outbound requests, the initial network release will necessitate an application to satisfy one of the following; open source code, stake greater than $10,000 USD worth of STR, or be in active use for > 6 months. As the network evolves, governance proposals can update the uncapped outbound requirements.

Note, the initial launch of the Stratosphere Network does not account for private operations, such as those proposed to run in Ekiden's trusted execution environment network [18].

4.6 Financial Cost Savings

Cost to Store 1 GB of Data

	ETH*	STR**	STR Cost Savings
100 Years	$4,000,000	$522.00	7,662x
10 Years	$4,000,000	$52.20	76,628x
1 Year	$4,000,000	$5.52	766,283x
1 Month	$4,000,000	$0.46	8,695,652x

* Assuming 1 byte = 5,000 gas, $4 / kilobyte at ETH price $160
** Assuming 10 nodes on Stratosphere Network, at a per node cost of 2x AWS ($0.023 / GB / Month)

Financial Cost Savings - Storage.

Data storage on the Ethereum network calculated at a rate of 1 byte cost of 5,000 gas, $4 / KB, at ETH price of $160. The theoretical cost of indefinitely storing **1 GB on Ethereum is $4,000,000**.

Stratosphere computation on AWS S3, under **ten (10) Stratosphere cloud nodes at 2x AWS costs** of $0.023 / GB / month. 1 month of 1 GB storage costs $0.46, a **financial savings of 8,695,652x less than ETH**. 1 year of 1 GB storage costs $5.52, a **financial savings of 766,283x less than ETH**.

Cost to Run 1 GB Size Database

	ETH*	STR**	STR Cost Savings
100 Years	$4,000,000	$2,760.00	1,449x
10 Years	$4,000,000	$276.00	14,492x
1 Year	$4,000,000	$27.60	144,927x
1 Month	$4,000,000	$2.30	1,739,130x

* Assuming 1 byte = 5,000 gas, $4 / kilobyte at ETH price $160
** Assuming 10 nodes on Stratosphere Network, at a per node cost of 2x AWS PostgreSQL General Purpose (SSD) storage ($0.115 / GB / Month)

Financial Cost Savings - Database.

Database on the Ethereum network calculated at a rate of 1 byte cost of 5,000 gas, $4 / KB, at ETH price of $160. The theoretical cost of indefinitely storing **1 GB on Ethereum is $4,000,000**.

Stratosphere computation on AWS PostgreSQL, under **ten (10) Stratosphere cloud nodes at 2x AWS costs** of $0.115 / GB / month. 1 month of 1 GB storage costs $2.30, a **financial savings of 1,739,130x less than ETH**. 1 year of 1 GB storage costs $27.60, a **financial savings of 144,927x less than ETH**.

Cost to Call a Smart Contract

	ETH	STR**	STR Cost Savings
1,000 Calls / sec	N/A	$0.33	Infinite
100 Calls / sec	N/A	$0.033	Infinite
10 Calls / sec	$5.00 - $50.00*	$0.00033	15,000x - 150,000x
1 Call / sec	$0.05 - $0.50	$0.000033	1,500x - 15,000x

* Assuming network congestion scales 10x with application increase in calls per second written on block
** Assuming 10 nodes on Stratosphere Network, at a per node cost of 2x AWS ($0.00001667 / GB-Second @ 512MB-200ms)

Financial Cost Savings - Computation.

Computation on the Ethereum network at a rate of **1 txn / sec** costs average 100,000 gas, equating to **$0.05 - $0.50 per compute-sec** depending on mempool size, price of ETH, and contract call. Scaled to **10 tx / sec**, network congestion can cause costs to logarithmically increase, in the range of **$5.00 - $50.00 per compute-sec**.

Stratosphere computation on AWS lambda, under **ten (10) Stratosphere cloud nodes at 2x AWS costs** of $0.00001667 / GB-Second @ 512MB-200ms. 1 txn / sec costs $0.000033 per compute-sec, a **financial savings of 1,500x - 15,000x less than ETH**. 10 txn / sec costs $0.00033 per compute-sec, a **financial savings of 15,000x - 150,000x less than ETH**.

5. APPLICATIONS

We categorize three high-level decentralized application networks: blockchain native, peer-to-peer, and multiparty.

5.1 Blockchain Native Networks

5.1.1 Decentralized Finance (DeFi). Expanding on the Ethereum ecosystem, we outline financial applications that necessitate more data and compute-intensive operations than current decentralized resources can provide.

5.1.1.1 Enterprise Payroll. Large scale, smart contract-based payroll would enable employees, for both global and remote teams, to be paid in a transparent manner.

Weekly payroll amounts per employee would be entered into a key value store database. As payroll changes, employee amounts can be seamlessly updated.

On payroll deposit, the escrow contract queries the database for payout amounts, pings the employer for final signature, and distributes designated payout amounts to each employee.

While control is owned by the employer, the employees are equipped with a transparent payroll process.

5.1.1.2 Decentralized Lending for Rental Properties. With the popularity of home vacation rentals, an application can be built to use a vacation rental protocol as a data stream. Consequently, lenders would know, in real-time, rental activity and conditions of the property.

The decentralized hosting of this applications database creates a network of trust between the lenders and the property status.

Since data entries can be frequently hashed on-chain, the public would be capable of verifying all previous events.

In addition, this protocol can work with the current vacation rental applications of today. Funds received from the vacation rental can be handled through a traditional escrow company. Hence loan repayments occur in real-time without the need to trust the borrower.

5.1.1.3 Off-Chain Trading. Many decentralized exchanges utilize an off-chain trading model to increase trading efficiency. On the Stratosphere Network, off-chain order books and trade executions can run across a distributed network of nodes.

As is the process for order books on the 0x Protocol [9], all orders would be entered into a simple database. EC2 instances would run an order matching process to connect market taker orders with market maker orders and send the matched orders to the main chain.

Unlike most platforms today, the entire off-chain orderbook system would take place across a distributed network of nodes with a decentrally maintained database. Additionally, both orders and transactions can be stored to another distributed, publicly verifiable database.

5.1.2 Non-Fungible Tokens (NFTs) for Business. Launching a high quantity of unique NFTs on Ethereum is impractically expensive.

On the Stratosphere Network, business-related NFTs, such as digital memberships, coupons, rewards points, and even emails, can be easily created with unique identifications at mass scale.

A simple PostgreSQL database can be used to store a ledger of application-specific NFTs. This structure enables NFTs to be cheaply created, transferred, and utilized, with unique identification and comprehensive metadata.

Businesses also benefit from the flexibility of such application-specific distributed databases. Initially, a business can use a general NFT database to launch its decentralized token program. As NFT utility expands, the organization can create a distributed NFT database specific to its ecosystem. In this way, custom rules can be constructed around transferability, ownership, and expiration.

5.1.3 Micro Betting / High-Speed Gambling. With a simple database and compute model, a micro betting platform can be built around live events. Betting networks on both private events, such as a race between two friends, or public events, such as on players streaming on Twitch, can be launched.

New bets, signed by the private key of both parties, would be sent to a "New Bet" EC2 instance. The compute instance would launch an escrow contract, enter the bet into the database, and wait for both members to deposit funds. On a fund deposit, the instance would use the oracle provided to subscribe to the event.

When an event completes, the EC2 instance immediately sends the information to the escrow contract for payout.

Both new bets and financial payouts happen at the speed of a centralized application, across a distributed network. Furthermore, such instances have the capacity to filter through large data streams, such as Twitch gaming streams.

5.1.4 Adaptive Oracles. Current oracle systems rely on mass adoption to support a range of data streams. With the Stratosphere network, oracles can be launched instantly. The developer simply deploys a compute function and selects the number of nodes. A function name and app id can then act as an oracle network endpoint.

5.2 Peer-to-Peer Networks

5.2.1 Digital Marketplaces. With powerful compute, database, and storage, scalable, decentralized online marketplaces can be seamlessly built to compete with centralized platforms.

5.2.1.1 Decentralized Ticket Exchange. A publicly accessible database would be at the center of a peer-to-peer ticket exchange.

Tickets entered into the database contain all necessary details, inputted as JSON data.

Queries are performed in the manner native to the database allowing all tickets to be retrieved at once or filtered through specific fields. This is far more efficient than the current query process in Ethereum and other blockchains.

Ticket purchases would operate on a simple lambda contract. The buyer would

enter the ticket ID and sign a preloaded payload. The signature would launch an escrow contract for the specific ticket, payment, and receiving wallets.

On deposit of exchanging assets, the escrow contract executes the preloaded swap. Transaction data can be added to another decentralized database to maintain a historical ledger.

Utilizing storage, the user-facing application could also be hosted across a distributed network of nodes furthering censorship resistance and decreasing platform risk.

5.2.1.2 Decentralized Music Streaming. Redistributing streaming profits from centralized applications to artists is a major promise of blockchain use in the music industry.

In one model, music listeners would own a membership NFT for monthly streaming. The NFT would be detected by the application interface to interact with the application. When the NFT is purchased, the receiving funds are deposited into an artist payout escrow contract.

In order for a song to be played, an mp3 file can be privately retrieved from the distributed storage nodes of the application. Along these lines, no song can be withheld by a single party.

Data points from the song streamed, such as user, stream length and song ID, would be entered in the application database.

In order for an artist to be paid transparently on stream, an escrow contract would be synched up to the streaming database.

On a monthly basis, the application would run a lambda function to allocate payment distribution per song. This information would be sent to the escrow contract for the final payout to the artists.

5.2.2 Physical Marketplaces. The peer-to-peer market of freelance, physical work has created a new employment stream for millions of self-employed freelancers.

However, these freelancers rely on centralized applications for all necessary operations ranging from compliance to payment. Decentralized versions of physical marketplaces will create a more transparent network providing freelancers with greater transparency in payout, autonomy, and fair treatment of their work.

5.2.2.1 Decentralized Ride Sharing. In a decentralized ride-sharing protocol, drivers can have full trust in fair treatment from the network as smart contracts dictate payment in real-time, while riders can fully verify drivers based on previous trips.

A basic version of a decentralized ride-sharing application can be architected with three Stratosphere cloud services - database, compute and storage - and the Stratosphere blockchain's native escrow contract.

To illustrate this, driver identity, trip history, and all other necessary data points would be stored in a standard PostgreSQL database.

Computation of route optimization, fee calculation, and trip completion would be deployed as lambda functions.

User facing applications would be stored as files in standard cloud servers and replicated across a network of hosting nodes.

Finally, driver payments would be made in real-time with the Stratosphere blockchain's native escrow contract.

On the Stratosphere network, all data points are publicly visible, in a manner similar to transactions on Etherscan.

5.2.2.2 Decentralized Food Delivery. The benefits of a decentralized food delivery protocol are similar to those of a decentralized ride-sharing protocol.

Deliverers can have full trust in fair treatment from the network, as smart contracts dictate payment in live time, and orderers can fully verify deliverers based on previous delivery history.

Additionally, restaurants can verify food payment and view a historical record of deliverer transactions.

To launch the network, three databases would need to be created for restaurants, orders, and deliverers. Restaurants are viewed by the system as a wallet address, with items and ratings entered in a PostgreSQL database. Customers can query this database for the desired restaurant or type of food.

When a customer creates an order, a lambda function is computed to calculate fees, launch an escrow contract, and post the order to the database. An EC2 instance is online to monitor a deposit payment in the escrow contract.

On payment deposit, the restaurant prepares the food and updates the database when ready. The restaurant and deliverer both sign a database update at pickup. Upon delivery, the customer signs the escrow contract, initiating the payment distribution to both the restaurant and deliverer.

5.2.3 Data-Intensive Networks. High bandwidth networks are infeasible to maintain, query, and update in the Ethereum blockchain. Data sizes for mass scale networks are in the order of petabytes. Stratosphere-built applications utilize the database services of major cloud providers, enabling massive database processing at baseline costs.

5.2.3.1 Decentralized Medical Data. Interoperability amongst medical systems and hospitals would be possible in a decentralized medical data protocol.

In an effort to create a secure electronic health or medical record accessible to any organization in the world, we envision the network utilizing a PGP encryption mechanism to encode patient data.

When medical data is entered into the database, both the organization and patient sign to encrypt the data. The public database acts as a ledger of encrypted data and is secured from major attack vectors of centralized electronic medical records like data breaches.

Patient data can be transferred to necessary organizations with a signature from the patient directly or through the signing organization.

During this process, data is verified to be correct and directly accessible to the patient through their Stratosphere wallet. The protocol also advances the shift of data autonomy and ownership from centralized data silos to the patient.

5.3.2.2 Decentralized Bidding Network. Real-time, transparent bidding has vast applications spanning from auction-based item marketplaces to real estate purchases.

In this specific protocol, assets listed for sale would be entered into a simple PostgreSQL database. New bids would be entered and updated in live time.

Bid finalization would trigger a "Finalize Bids" lambda function. The function would run a database query to select the highest bidder, launch an escrow contract, and deploy an EC2 instance for order tracking.

On paid escrow deposit, the EC2 instance would begin monitoring for the successful delivery of the sold item. An outside data stream, specific to the item would be built into the tracking function.

For digital items, an NFT can be utilized to represent a delivery receipt. For physical items, tracking information from UPS can be entered into the decentralized database through an adaptive oracle.

5.3 Multiparty Networks

5.3.1 Informational Trust Networks. Digital communication and information consumption is prime for disruption. Decentralized protocols can be used to create a more open and truthful internet.

5.3.1.1 Decentralized Social Media. Novel social media platforms can be built for the general public as censorship-resistant vehicles for open communication.

In a decentralized social network, the value would derive from usage - not advertisements. Thus, protocols may work to create a transparent system of the content display without compromising quality for monetary gains.

The minimal version of this network could be built with a Graph Database and various lambda functions.

The database would enable semantic queries, such as those of users, likes, or posts, in addition, to entries of relational data points.

Lambda functions would compute all necessary operations, such as the population of the newsfeed and notification of birthdays.

5.3.1.2 Decentralized News Publication. On focusing to create a more transparent news outlet, we propose an idea for a decentralized media publication.

The news outlet would be governed by a set of validators and allow any viable reporter to submit an article to the network.

Upon article submission, the validator set would verify the truthfulness of articles and op-eds. Articles that surpass a specific threshold are published on the ledger.

In a fashion similar to Etherscan, news media statements, publications, and reporters can be queried and linked.

Using this protocol, links to news articles could be used as a trusted ledger. For example, such a link can be included in a tweet regarding a previous event.

5.3.1.3 Validation Network on Government Statements and Actions. Operating in a similar fashion to a decentralized news publication, a government validation protocol would verify the truthfulness of statements made by government officials.

Consequently, a truthfulness ranking of public officials could be created in a reliable, transparent manner. Similar to Etherscan, all statements would be

accessible from a decentralized, verified, and immutable database.

Following our example, the decision to include such a reference in a tweet could increase transparency and trust in the social media ecosystem.

5.3.1.4 Charity Transparency. Similar to the decentralized payroll, decentralized charity applications would be built to process ongoing operations and payments.

In one model, the charity must list the specific amount paid to administrators, employees, and vendors. Each member in the database contains a wallet associated with their account.

On each month-end, vendors submit their final bills updating payout amounts on the database. Then, the application's compute operation triggers a "Payout" lambda function. This initiates the distribution of funds from the donation wallet to the receiving parties.

Since organizational complexities escalate exponentially with size, it would be difficult to orchestrate the various moving pieces of a decentralized charity on Ethereum. The Stratosphere Network can also create a transparent ledger around all non-financial information related to ongoing charity activities.

5.3.1.5 Decentralized Compliance. The proposed compliance protocol includes both Know Your Customer (KYC) and Anti-Money Laundering (AML) processes.

Standard practices necessitate KYC information to be easily accessed and updated. This process is made seamless with the Stratosphere database services. KYC data can be encrypted and only revealed or updated when necessary.

Anti-Money Laundering detection software can be open-sourced and run through the Stratosphere compute services. Hence, compliance can be made available to decentralized applications at the application's onset.

5.3.2 Decentralized Service Networks. In models where a service provider must be trusted to fulfill various tasks, decentralized networks offer an increase in transparency and trust.

5.3.2.1 Decentralized Travel Agent. Today, travelers pay the agent upfront and expect services to be fully arranged. However, this is not always the case [10].

The travel agent application would run on a simple PostgreSQL database, where the agent enters all arranged services for a specific trip. The data for the trip is encrypted by both the agent and customer wallets.

An escrow contract is deployed to hold customer funds. Payment distribution, eg. the amount paid to each service provider, is pre-committed to the escrow contract.

Depending on the service, payments may be made ahead of time or immediately prior to the event. The customer can sufficiently trust that all trip services are covered.

5.3.2.2 Decentralized Vacation Rentals. Without needing to adhere to the conditions of a trusted middleman, homeowners would be in greater charge of their vacation rental and resistant to platform censorship.

The protocol would be architected in the same fashion as current centralized vacation rental applications.

A standard database would be used to log all entries from property specifics to the history of previous stays. Both the host and the guest would be processed through a KYC mechanism.

EC2 and lambda functions would execute all following compute operations, such as bookings, queries, preferences, and ratings.

Payment for the stay would be sent to an escrow contract beforehand assuring the host of payment. At the end of the stay, the guest would have a time period to refute the stay. Otherwise, the escrow payment will be sent.

We envision this application to operate as a protocol, similar to the design of Compound Finance [11]. The regulatory compliance needed is different per geographic area resulting in a multitude of user-facing interfaces that would need to be built to accommodate them.

5.3.2.3 Decentralized Freelance Market. The freelance protocol would have a similar architecture to the decentralized vacation rental application.

The database would store all information related to freelancers, proposals, and track history. Computation resources would execute all standard functions, such as selecting a freelancer.

The unique difference between this and the vacation rental architecture is the splitting of funding and verification of milestone payouts.

While the payment for the proposal is deposited into a smart contract beforehand, the freelancer will only receive portions at a time. So, both the freelancer and company must agree on milestone completion for the escrow contract to disburse designated funds.

5.3.3 Decentralized Computation. In the Stratosphere network, computation resources can theoretically expand to the bandwidth of all centralized cloud service providers combined. In practicality, this large bandwidth can support a decentralized model of the many innovative, resource-intensive applications.

5.3.3.1 Decentralized AI / ML. Massive data collections and feature specific computations are needed for a decentralized AI to perform at full capacity.

The opportunity to create such an operation is extremely powerful for running global networks at scale. An open-sourced, publicly verifiable artificial intelligence furthers the mission to ensure artificial general intelligence benefits all of humanity, a view shared by OpenAI.

Database construction within a decentralized AI model would likely utilize BigTable or a large storage instance. This data could be either publicly accessible or privately encrypted.

The specific computation algorithm used can be hand-picked from the vast offerings from cloud service providers. For example, a k-Nearest Neighbors application may choose between Amazon’s SageMaker or Google's TensorFlow. The application could also run on both service providers.

For greater distribution of AI, instances can be run across many nodes. The model proposed functions properly as each node itself is specifically suited for this massive computation of data.

5.3.3.2 Decentralized Gaming. In a decentralized network, video games would be able to operate in an open, censorship-resistant environment.

The major constraint of gaming architecture is the maintenance of a shared state. Database and computation capacity must be efficient for all global events to update in real-time. This necessity is magnified in the decentralized model as participant updates are streamed to all nodes in the network.

Utilizing a real-time database, such as Firebase's Cloud Firestore, and speed-focused compute resources, such as EC2 zid instances, a decentralized video game could be built to service a large, open community of gamers.

Decentralized autonomous organizations (DAOs) are an interesting addition to this model of decentralized gaming. The DAO would be governed by the community to set game rules and deal with disputes.

5.3.3.3 Decentralized VR / AR. Similar to decentralized gaming, large scale resources are needed to process decentralized VR or AR.

However, key differences between VR / AR and video game architecture include (1) user-entered datasets and (2) lack of user-facing interfaces.

In real-world VR / AR, users can create augmented structures, designs, maps, etc. These items need to be stored in a more complex database, potentially a combination of Graph and PostgreSQL.

Unlike gaming, VR / AR environments can be novel and frequently changing. While the decentralized network would function as an underlying protocol, user-facing interfaces would likely be more flexible and created by end application developers.

6. CRYPTOECONOMICS

The Stratosphere Network acts as a **networking layer between the end developer and the $230B cloud market**, an industry expected to grow to $500B by 2023 [25]. To facilitate operations, Stratosphere is built with two native tokens, STB and STR.

6.1 STB: Stratosphere Stablecoin

STB will function as the network's native stablecoin. STB is a wrapped version of USD Coin (USDC), enabling an immediate fiat on-ramp into the Stratosphere Network.

We initially utilize USDC, as opposed to other stablecoins, as USDC on-ramps offer a no-fee purchase. Thus, $1 USD will equate to 1 STB. The inherent risk of sole reliance on USDC is the trust placed in the token co-creators (Coinbase and Circle), the bank account of residing fiat, and the governing body of USDC. However, we view the security provided by USDC as highest amongst stablecoins to-date. As the Stratosphere network grows, additional stablecoins can be included to support STB.

STB can be created by purchasing USDC through secure, one-to-one on-ramps, such as Coinbase and Circle, and depositing USDC into the STB Minter contract on Ethereum. The receiving Stratosphere wallet address is entered in the STB mint transaction.

Stratosphere contains a native contract to monitor USDC deposits. The contract mints the reciprocal STB on the Stratosphere Network. The conversion of STB to USDC works in a similar manner except in the reverse order. As a result, network participants can enter or leave the Stratosphere Network at discretion.

6.2 STR: Stratosphere Native Token

The Stratosphere Native Token (STR) acts as the native protocol token. The token performs functions in node pre-payment, governance, staking, and app mining.

6.3 Node Payment

Nodes are paid for performing cloud service operations. Payments are split into cost-basis and profit margins. The initial payment rate set by the network is 2x cost of the cloud service rate.

6.3.1 Cost Basis in STB

Costs basis for cloud computing is set to be fully covered in STB. In addition, +20% of costs are proposed to be paid in STB. This +20% will be divided into two distributions, 10% sent to the node as STB profit and 10% sent to the Stratosphere Network Swap Reserve (described below).

6.3.2 Profit Margin in STR

Profit margins for cloud computing, past the first +20% above costs, are to be paid in STR. The node can set this amount to be received.

Received STR is divided into three sections. 50% is paid to the node as profit for services. 25% is sent to Network Support (described below). 25% is burned (described below).

Figure 4: Payment % Summary.

6.3.3 Payment Example

Let's use the example of a node that entails a cost worth $100 USD. At the payment rate of 2x costs, 120 STB and $80 USD worth of STR (calculated at the time of prepayment deposit) would be withdrawn from the prepaid deposits.

The node would receive 100 STB to cover costs, as well as 10 STB and $40 USD worth of STR as profit.

The remaining 10 STB would be sent to the Swap Reserve (described below), while $20 USD worth of STR would be sent to Network Support (described below) and $20 USD worth of STR would be burned (described below).

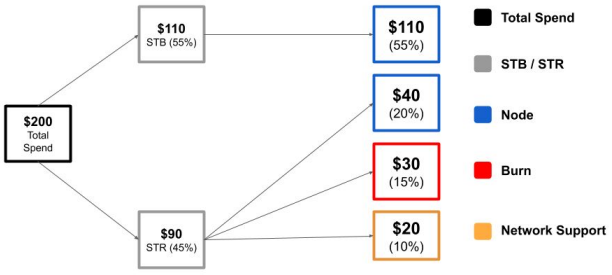

Figure 3: Payment Example Summary Diagram.

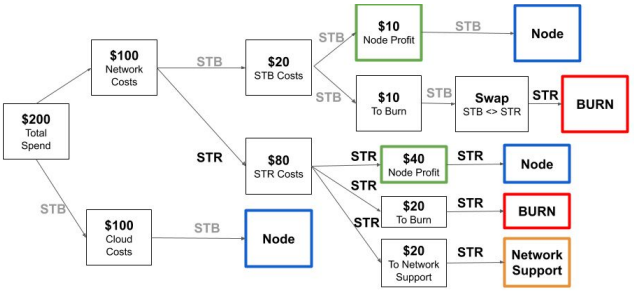

Figure 4: Payment Example Detailed Diagram.

6.4 Burn Model

The Stratosphere blockchain will contain a native burn contract, in which any STR sent will be transferred to the wallet address 0x0.

We introduce two burn models into the STR token ecosystem.

The first burn is administered on profit margin payments to nodes. We propose a burn of total spend at **15%.** The burn % is calculated by both the first STR burn (directly from STR spend) and the second STR burn (through the swap reserve contract). This number is adjustable through a super-majority governance vote.

As stated, the second burn is administered on the Swap Reserve (described below). STR from the Swap Reserve Contract is automatically sent to the burn contract.

6.5 Network Support

The aim of the Network Support is to continue development of the Stratosphere Network and ecosystem.

Network Support is initially proposed at 10% of the total payment, in the form of STR. This number is adjustable through a super-majority governance vote.

Community governance will be utilized for both allocations of Network Support and approval of specific projects.

6.5.1 Core Development

Core development encompasses work on the Stratosphere network, protocol, and blockchain. Team members employed or contracted by the foundation will receive STR in exchange for work produced.

6.5.2 App Mining

To incentivize app developers [12], we introduce an app mining program. A pool of STR is accrued each month to reward developers for their latest releases.

STR holders in the Stratosphere Network can vote on the most popular apps by latest release for the month. At voting conclusion, a leaderboard is established and STR is distributed proportionally to the developers.

6.5.3 Bug Bounty

Continued improvement of network security is a must in the Stratosphere Network. We implement a bug bounty to promote vulnerability discovery and white hat hacking.

As more cloud services are integrated into the network, the need for such bug bounties will increase over time.

6.6 Stratoswap

If not built by the community, the Stratosphere Network will launch a native token **order book and liquidity pool**. The initial exchange would allow for both limit and market orders for the STR / STB pair.

The exchange would operate similar to a 0x order book, where an order book is hosted off-chain and final execution occurs on-chain [9]. Market makers may contribute STR or STB to the liquidity pool, in exchange for fees from exchange commission.

6.7 Swap Reserve Contract

The Swap Reserve is a native smart contract on the Stratosphere blockchain that shares its liquidity with Stratoswap. The contract serves one main operation, sectioned into three main steps: deposit STB, place a market order for STR with STB on Stratoswap, auto-send STR to the burn contract.

Depositing STB is triggered when the proportional node profit margin of STB is sent into the Swap Reserve contract.

After the STB deposit, the Stratoswap market order is subsequently called and this exchanges the deposited STB for STR from Stratoswap.

STR received by the Swap Reserve triggers an automatic transfer of the STR to the burn contact.

As a whole, the Swap Reserve mechanism provides network benefits in the form of increased liquidity and induced burn.

6.8 Governance

The two layer structure of the Stratosphere Network necessitates a governance structure to both maintain overall state over long periods of time, while also quickly adapting to immediate threats and opportunities.

Thus, we propose a two layer governance structure, consisting of a **general governing body** and a **subcommittee layer**, similar to the operations of the United States Federal Government.

Major Stratosphere blockchain and protocol proposals will filter through a conservative amendment system, based on the governance structure of the Tezos Blockchain [24]. The four step process necessitates amendments to be pre-selected, approved, adopted and approved in tested, and integrated into the underlying protocol. Each step maintains an examination and voting period of 90 days. Any member of the Stratosphere community can submit a proposal through their delegated validator.

Core network parameters may necessitate more immediate change, thus we follow to the parameter upgrade model built by Cosmos on Tendermint [8]. As these upgrades affect all network members, each Validators must vote on core parameter upgrades, or risk time-out penalization.

During the voting period, each STR token represents one pro-rata vote in the ecosystem.

Core network parameters include:

STB Profit Margin %: Percent of STB costs to be paid as additional profit margin. The initial rate is set at 10%.

STB Node Profit %: Percent of STB Profit Margin % to be sent to the node. The initial rate is set at 5%.

STB Reserve Swap %: Percent of STB Profit Margin % to be sent to the Reserve Swap contract. The initial rate is set at 5%.

STR Profit Margin %: Percent of STB costs to be paid in STR as additional profit margin. The initial rate is set at 40%.

STR Node Profit %: Percent of STR Profit Margin % to be sent to the node. The initial rate is set at 20%.

STR Burn %: Percent of STR Profit Margin % to be sent to the burn contract. The initial rate is set at 10%.

STR Network Support %: Percent of STR Profit Margin % to be sent to the Network Support. The initial rate is set at 10%.

Subcommittees are the most flexible and adaptable governance bodies of the Stratosphere Network. Subcommittees can create allowed lists, perform slashing operations, set minimums on specific metrics, etc. The flexible governance mechanisms of subcommittees for proposal amendments are highly beneficial for adaptive architecture and continuous development, a governance model demonstrated by 0x Protocol [9].

Initial subcommittee's to be launched include; Network Support Committee, Flagged Activity List Committee, Insurance Pool Payouts Committee, Complaint Network Committee, App Mining Voting Committee. The organizations may act similar to Decentralized Autonomous Organizations (DAOs) on Ethereum [2], however they must abide to overarching rules established by the main governing body.

The Network Support Committee will initially be in charge of STR allocation to core development,

app mining, and bug bounty. Community voting may also be utilized to vote on specific projects or proposals within each category.

The Flagged Application List Committee performs a crucial cybersecurity role in the network. As both cyberattacks and development opportunities are ever evolving, members will be able to freely propose new flagged activities, describe how to handle each, and associate potential bounties for specific behaviors. Flagged activities include cyberattacks, inappropriate application usage, geographic specific regulatory disclosures, non-deterministic functions, one-time executions, etc.

The App Mining Committee will initially have a monthly voting process. The aim is to vote on which apps had the most popular feature releases in the previous month. Following along these lines, the app mining pool will incentive developers to continue iterating on product or protocol development.

The Complaint Network Committee will review and dictate complaints from either applications or nodes. For example, an app may submit a complaint on a node, with evidence and a MinDeposit. The validators verify the complaint. If consensus is reached on the complaint, then the complaint is added to node's reputation on-chain and the app receives 80% of MinDeposit back. If consensus on complaint reject, then 80% of MinDeposit sent to reserve pool. 20% of MinDeposit pays Validators to perform their work.

Launching a subcommittee is trivial, simply submitting a JSON request to the Stratosphere RPC with subcommittee information. However, subcommittees remain inactive until greater than 1% of all STR is delegated to the subcommittee. An STR token can only be delegated to one subcommittee at a time.

6.9 Staking

Validator rewards are distributed per staking period on an inflation rate set by the network. The preliminary initial inflation rate will be set at 11% yearly STR inflation with a logarithmically decreasing inflation yield curve.

Non-validators can participate in rewards by delegating STR stake to live validators.

Both applications and nodes must participate in staking. The specific staking quantity will be dictated per party, based on various reputation, operation, and flagged variables.

Subcommittee requests may also utilize staking as a necessity for submission. For example, an application must stake X amount of STR to submit a complaint.

6.10 Slashing

Slashing as a penalty for validators follows the same manner as validator slashing in Tendermint BFT. As stated in the Cosmos white paper,

“There must be some penalty imposed on the validators for any intentional or unintentional deviation from the sanctioned protocol. Some evidence is immediately admissible, such as a double-sign at the same height and round, or a violation of "prevote-the-lock" (a rule of the Tendermint consensus protocol). Such evidence will result in the validator losing its good standing and its bonded atoms as well its proportionate share of tokens in the reserve pool -- collectively called its "stake" -- will get slashed.” [8]

We propose an exponentially time-weighted slash rate to systematically penalize bad actors, without overly penalizing isolated errors. An initial in a two year time window will begin at 5%, and increase by a factor of 2x per subsequent slash within the time window. Slash penalty inversely decreases by the same factor of 2x per time window expiration.

Network based slashing is dictated by the relevant subcommittee and specific to the scenario. For example, the amount of STR slashed from a node found of cheating will be voted upon by relevant subcommittee representing Validators.

7. Conclusion

The Stratosphere network creates a distributed layer of transparency atop of the current cloud ecosystem. Within a decentralized application, major cloud services, such as computation, database, and storage, can be performed quickly, securely and cost-effectively, while maintaining the major trust benefits of decentralization.

The Stratosphere network is constructed with two innate layers: the blockchain layer and the application layer. The former is a proof of stake Tendermint-based blockchain, secured by a network of validators. The latter is a distributed network of cloud service nodes that run application-specific services in parallel and secured through the Trusted Function primitive.

We propose the Stratosphere blockchain in an effort to unlock the next generation of decentralized applications, in which trusted, immutable, publicly verifiable protocols are built for scale.

References

[1] Reza Shiftehfar. *Uber's Big Data Platform: 100+ Petabytes with Minute Latency.* Oct 2018. URL: https://eng.uber.com/uber-big-data-platform/

[2] Vitalik Buterin. Ethereum White Paper *A NEXT GENERATION SMART CONTRACT & DECENTRALIZED APPLICATION PLATFORM* URL: http://blockchainlab.com/pdf/Ethereum_white_paper-a_next_generation_smart_contract_and_decentralized_application_platform-vitalik-buterin.pdf

[3] Ethereum Name Service (ENS). URL: https://ens.domains/

[4] Amazon Web Services (AWS). URL: https://aws.amazon.com/

[5] Microsoft Azure Cloud Computing Platform. URL: https://azure.microsoft.com/en-us/

[6] Google Cloud Platform (GCP). URL: https://cloud.google.com/

[7] Ethan Buchman. *Tendermint: Byzantine Fault Tolerance in the Age of Blockchains.* Jun 2016. URL: https://allquantor.at/blockchainbib/pdf/buchman2016tendermint.pdf

[8] Jae Kwon, Ethan Buchman. URL: https://cosmos.network/resources/whitepaper

[9] Will Warren, Amir Bandeali. *0x: An open protocol for decentralized exchange on the Ethereum blockchain* Feb 2017. URL: https://0x.org/pdfs/0x_white_paper.pdf

[10] Stacey Leasca. *Tour Operator Thomas Cook Ceases Operations, Leaving 600,000 People Stranded Across the Globe.* Sep 2019. URL: https://www.travelandleisure.com/travel-news/thomas-cook-ceases-operations-leaves-travelers-stranded

[11] Robert Leshner, Geoffrey Hayes. *Compound: The Money Market Protocol.* Feb 2019. URL: https://compound.finance/documents/Compound.Whitepaper.pdf

[12] Blockstack Token LLC. *Blockstack Regulation A SEC Filing.* Apr 2019. URL: https://www.sec.gov/Archives/edgar/data/1719379/000110465919020748/a18-15736_1partiiandiii.html

[13] Robert Greenfield IV. *Reputation on Blockchain.* Nov 2017. URL: https://medium.com/@robertgreenfieldiv/reputation-on-the-blockchain-624947b36897

[14] Zulfikar Ramzan, Vijay Seshadri, and Carey Nachenberg. *Reputation Based Security, An Analysis on Real World Effectiveness.* URL: https://www.symantec.com/content/dam/symantec/docs/security-center/white-papers/security-response-reputation-based-security-10-en.pdf

[15] Alice Cheng, Eric Friedman. *Sybil Proof Reputation Mechanisms.* URL: http://www.eecs.harvard.edu/cs286r/courses/fall08/files/paper-CheFri.pdf

[16] Jeff Melnick. *Top 10 Most Common Types of Cyber Attacks.* May 2018. URL: https://blog.netwrix.com/2018/05/15/top-10-most-common-types-of-cyber-attacks/

[17] US Signal. *Protecting Against Cyber Attacks with Rate-Limiting.* Jan 2019. URL: https://ussignal.com/blog/protect-against-cyber-attacks-with-rate-limiting

[18] Raymond Cheng, Fan Zhang, Jernej Kos, Warren He, Nicholas Hynes, Noah Johnson, Ari Juels, Andrew Miller, Dawn Song. *Ekiden: A Platform for Confidentiality-Preserving, Trustworthy, and Performant Smart Contracts.* URL: https://arxiv.org/pdf/1804.05141.pdf

[19] Noah Ruderman. *How Inefficient Are Dapps?*. Sep 2018. URL: https://medium.com/coinmonks/how-inefficient-are-dapps-c18062c80a71

[20] Allen Day. *Building Hybrid Blockchain/Cloud Applications on Ethereum and Google Cloud.* Jun 2019. URL: https://cloud.google.com/blog/products/data-analytics/building-hybrid-blockchain-cloud-applications-with-ethereum-and-google-cloud

[21] Evgeny Ponomarev. *Dapp Survey Results 2019.* Jan 2019. URL: https://medium.com/fluence-network/dapp-survey-results-2019-a04373db6452

[22] Overclock Labs. *Akash Network: Decentralized Cloud Infrastructure Marketplace.* May 2018. URL: https://akash.network/static/akash-position.pdf

[23] Steve Ellis, Ari Juels† , and Sergey Nazarov. *Chainlink. A Decentralized Oracle Network.* Sep 2017. URL: https://link.smartcontract.com/whitepaper

[24] L.M Goodman. *Tezos -- A Self Amending Crypto Ledger.* Sep 2014. URL: https://tezos.com/static/white_paper-2dc8c02267a8fb86bd67a108199441bf.pdf

[25] Michael Shirer, Eileen Smith, IDC. *Worldwide Public Cloud Services Spending Will More Than Double by 2023, According to IDC.* Jul 2019. URL: https://www.idc.com/getdoc.jsp?containerId=prUS45340719

[26] David Canellis. *More than 60% of Ethereum nodes run in the cloud, mostly on Amazon Web Services.* Sep 2019. URL: https://thenextweb.com/hardfork/2019/09/23/ethereum-nodes-cloud-services-amazon-web-services-blockchain-hosted-decentralization/