# Draft: Proximum ∗

Chris Walker
chris@ckwalker.com

**Abstract**

Proximum aims to augment physics with trustless computation. It is the first blockchain with a consensus mechanism and fractal network structure mapped onto spacetime to maximize scale and decentralization. It relies on the speed of light for a new trustless proof of proximity, uses zero-knowledge proofs to compress computation at each network scale, and enables new location-dependent blockchain applications such as truly local flatcoins.

**Disclaimer:** *This is an incomplete draft document. We welcome comments, critiques, and improvements. Nothing in this document is a promise or guarantee of any kind. Sometimes a paper is just a paper.*

## Contents

# 1 Introduction

Blockchains are best modeled as virtual physics: they aim to augment base physics with additional functions corresponding to universal physical laws and additional data corresponding to local physical structures, where all participants can reach consensus on the state transitions of relevant structures through successive application of the laws. Any application which does not need these properties does not need a blockchain. When one imagines a network which perfectly meets traditional blockchain goals of decentralization, security, speed, and scalability, it approximates a physical principle: universal, immutable, and perfectly parallel. But there is plenty of room to improve the fidelity with which blockchains augment physics.

## Decentralization and Security

In general, Proof of Stake blockchains backed by economic incentives offer security bounded by the value at risk and are subject to the wealth-centralizing effects of staking rewards. Many networks (e.g. delegated proof of stake systems) rely on small sets of participants to determine consensus and are thus better modeled as central planning committees rather than physical principles–we will not consider them further. Proof of Work blockchains reduce this wealth concentration effect because miners must pay for the resource expended during mining, but this ongoing cost of security is unlikely to be sustainable in the absense of inflationary mining rewards[28]. Many mining systems which require significant resource consumption are subject to the centralization caused by economies of scale and are subject to potential coercion from local authorities.

Let us imagine an ideal solution: it would be tethered to physical rather than economic principles, just like extant Proof of Work systems, but it would require a small ongoing cost rather than a large one (as an analogy, defending a city with a wall is more cost-effective in the long run than maintaining a large standing army). In addition, it would be extremely decentralized to minimize the risk from coercive jurisdictions.

## Latency and Scale

Current blockchains have a high minimum latency because they must reach consensus globally. In addition, they have a low minimum scale because their consensus algorithms and incentive models are not suitable for distant participants, e.g. Martian colonies.

An ideal network would be able to achieve both extremely low latencies through locally defined consensus and extremely high scale such as the solar system.

## Economics

The primary reason to augment physics with blockchain is to construct new economic systems. The past decade has shown that Bitcoin's use as a store of value and tokenized representations of the US Dollar, such as USDC and USDT, used are the most important specific uses for blockchains. But blockchains have yet to construct a widely accepted asset which serves as store of value, unit of account, and medium of exchange. While tokenized dollars are more functional than legacy dollars, they by definition cannot free us from the ills of the fiat currency regime. Because Bitcoin has a perfectly inelastic supply curve and the demand for money varies significantly over time as economic conditions change, Bitcoin can never reach a stable price. An ideal cryptoeconomic system would support the creation of *many* flatcoins which maintain a stable price with respect to themselves and a reference basket of prices, and which can be cost-effectively used at the scale of day-to-day commerce by regular people.

## Physical analogies

We can look to the natural world for suggestions on how to build a network that better achieves these goals.

**Universality** Filling near-Earth spacetime with a dense fabric of independent nodes is a useful approximation of a universal law. In blockchain terms, consensus may be defined as the assent of a majority of occupied spacetime, forming a less-centralized and more cost-effective proof of work than existing blockchains.

**Local Parallelism and Fractal Structure** Physics appears to scale through local computation and speed limits on causal propagation. Blockchains can scale through parallelism, i.e. defining local regions with bounded information propagation between regions enabling parallel computation.

**Fractal Structure** Natural phenomena often use fractal structures to maximize throughput, such as the branching of trees (maximizing absorbed sunlight per trunk area) or the structure of lungs (diffusion per volume)[20]. Arranging local blockchain regions in a fractal structure mapped directly to spacetime is an intuitive way to maximize throughput and minimize latency.

Proximum is motivated by these intutions, but one can evaluate the network without accepting them.

## 2 Foundations

The Proximum network consists of a set of nodes $n \in$ Nodes participating in the Proximum protocol. Each node $n$ holds a unique keypair $K_n = (K_{n_{\text{public}}}, K_{n_{\text{private}}})$ and is represented by a unique address $A_n$ which can be derived from each public key $A_n = f(K_{\text{public}})$.

The network comprises a set of subnets, each of which includes a public registry mapping an address for each node within the subnet to the corresponding public key and an asserted location $\mathbf{r}$ in $\mathbb{R}^3$:

$$A_n \mapsto \{K_{n_{\text{public}}}, \mathbf{r}_n\}.$$

Each block of transactions created within subnet $N^i_{I_1}$ is compressed into a single transaction which is verified by the parent subnet $N^{i-1}_{I_2}$.

Furthermore, Proximum depends on the following principles and assumptions: see the Appendix for discussion of each.

### Byzantine fault tolerance

The network is Byzantine-fault-tolerant: given $2m+1$ honest nodes following the protocol correctly, these nodes can use an algorithm to reach consensus on the true network state despite up to $m$ adversarial or faulty nodes. Third party users interacting with Proximum do not need to participate in the protocol directly: to verify the state of Proximum, a third party user $u$ must trust only one honest node $n$.

### One-way hash functions

A unique fingerprint or hash $h$ can be created for any message $m$: $h = H(m)$.

### Verifiable delay function (VDF)

A hash function $H$ recursed $n$ times can be used to verify a minimum delay between two events $x$ and $y$ based on the time $\tau$ required to compute each hash. This hash function can be used as an approximate clock: $(y, \pi) = \text{VDF}^n_{\text{Eval}}(x) = H^n(x)$. Also note that additional data can be interleaved into this hash function.

### Digital signatures

A node $n$ holding a known keypair $K$ can attest to a message $x$ by adding an unforgeable signature $S$ to create the signed message $M_{S_n}(x)$ that can be verified by any observer.

### Zero-knowledge proofs

A prover $P$ can generate a proof $\Pi$ of an underlying computation $C$ which can be checked by a verifier $V$. Zero-knowledge proofs enable computation performed by one party to be verified by another party without revealing any further information about that computation.

### Spatial indexing

Earth's surface may be tiled into roughly hexagonal cells using the H3 index at a variety of resolutions, where $I_n$ denotes the label for a specific cell at resolution $n$. Statistics for these cells are shown in Table 2.

**Speed of light**

Information cannot be transmitted faster than the speed of light.

**Extended Kalman Filters**

The Extended Kalman Filter (EKF) is a standard method for estimating the state of a system with a nonlinear state transition function and non-Gaussian noise. In Proximum, it is used to estimate the position $\hat{\mathbf{r}}_n$ of each node $n$ based on its asserted location $\mathbf{r}_n$ and measurements consisting of trustless time of flight of messages between nodes.

This estimated node position is used within Proximum to determine the assent of the majority of spacetime and can be relied upon by third parties to estimate the position of users interacting with the Proximum network.

## 3 Proximity

Proximum nodes can verify mutual proximity in a trustless manner. Unlike traditional time-of-flight calculations [26], nodes cannot rely on each other to honestly report locations or event timing. A simple proof of proximity or PoP algorithm, denoted $\bar{d}(n_1, n_2)$, allows any node $n_1$ to bound the spatial distance to any counterparty.

### Two-way proximity

Node $n_A$ running a VDF beginning with the value $\text{VDF}_{\text{Eval}}(x_0)$ at $t_0$ can verify proximity to a counterparty $n_B$ using these steps, as illustrated in Figure 3.

1. Node $A$ signs a message $m_1 = M_{S_A}(y_i)$ including the most recent VDF output $y_i$ using keypair $K_A$ and folds this message into its own VDF: $y_{i+1} = H(y_i, m_1)$.
2. Node $A$ transmits this signed message $m_1$ to node $B$.
3. Node $B$ receives the message and adds its signature: $m_2 = M_{S_B}(m_1)$.
4. Node $B$ transmits this signed message $m_2$ back to node $A$.
5. Node $A$ receives the message prior to the next VDF output $y_k$ and folds $m_2$ into the VDF: $y_k = H(y_{k-1}, m_2)$.

Node $n_A$ can then bound the distance $d_{n_A,n_B}$ to node $n_B$ using the following inequality:

$$d_{n_A,n_B} < \bar{d}(n_A, n_B) = \frac{c\tau(k-i)}{2}$$

where $c\tau$ represents the distance light travels per VDF iteration, and $k - i$ represents the number of VDF iterations between finding $y_i$ and finding $y_k$. Note that node $n_B$ does not need to run the VDF to participate in this proof, and thus can be replaced by any user interacting with Proximum.

Node $n_A$ does not need to extend any trust to its counterparty because $m_2$ includes the pseudorandom value $y_i$ and node $n_A$'s signature. Therefore it cannot have been generated in advance by $n_B$. Obviously node $n_B$ can follow similar steps to verify proximity to $n_A$. This forms the basis of Proximum's trustless proof of proximity.

While node $A$ can trustlessly prove the proximity of $B$, third parties cannot directly rely on $m_1, m_2, \dots$ as a trustless proof because $A$ and $B$ may have colluded by signing messages containing stale VDF values to understate their distance, etc.

### N-way position estimation

Let $N$ be the set of Proximum nodes, $I$ be a spatial region defined by location $\mathbf{r}_I$ and threshold distance $\delta$, $N_{\mathbf{r}} \subseteq N$ be the nodes actually within this region, and let $N'_{\mathbf{r}} \subseteq N$ be the nodes *claiming* to occupy this region. How can any observer accurately estimate which nodes, $\hat{N}_{\mathbf{r}} \subseteq N$, are actually within this region?

Consider the following brute force solution.

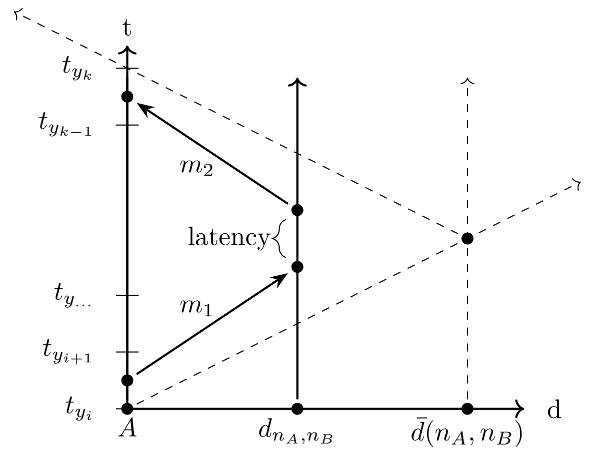

Figure 1: Node B can trustlessly demonstrate proximity to node A. The true distance $d_{n_A,n_B}$ must be less than $\bar{d}(n_A, n_B)$ due to latency and message propagation speed. Note that the dotted diagonal lines illustrate sections of Minkowski lightcones for A and B.

1. Every node in $N$ asserts a position $\mathbf{r}'$ and a resolution $l'_n$ describing the region it claims to occupy.

2. Each node completes a two-way proximity proof with every other node in $N$ and signs its proof. Yes, this is an $O(n^2)$ solution, bear with us.

3. One solver computes a position estimate $\hat{\mathbf{r}}$ for every node using an Extended Kalman Filter (EKF) based on these proximity proofs and asserted positions $\mathbf{r}'$. Note that an EKF also provides a measure of uncertainty for each position estimate.

4. The solver removes nodes with a probability below $\delta$ of being located within their asserted region, $P(|\hat{\mathbf{r}} - \mathbf{r}'| < \frac{l_n}{2}) < \delta$, and those which signed unreasonable proofs of proximity, to find the consensus set $\hat{N}$.

5. A zero-knowledge proof of consensus set membership is created by the solver and broadcast to network participants (remember, this is a brute force solution)

Now any node or observer can trustlessly verify that a majority of nodes within a region have reached consensus on a state transition. In practice, we can make several simplifying changes: the EKF can update position estimates based on any number of two-way proofs of proximity rather than requiring all nodes to participate, seperate EKFs can be run per subnet to minimize the size of the node states within each EKF, and each EKF solution can be compressed into a zero-knowledge proof only when specific confidence intervals have been reached on node positions.

We also must consider the fact that the VDF values will drift slightly between participants, and that nodes should not be able to sybil attack the proximity results. Consider the revised algorithm below.

1. Node $n$ reaches a qualifying VDF output $y_i$, e.g. where the $i \bmod \pi_{\text{block}} = 0$ or $y_i < \pi_{\text{block}}$ for some parameter $\pi_{\text{block}}$.

2. Node $n$ completes a two-way proximity proof with pseudo-randomly selected nodes in other regions.

3. State proofs are found and broadcast to the network

This n-way position estimation algorithm forms the basis of Proximum's consensus mechanism.

## Latency

The resolution of proximity proofs are limited by the latency of typical computers: light travels 9,000,000 meters during a reasonable 30 ms response time, so time of flight proximity proofs will provide little value on regular networks[29]. Purpose-built systems can achieve much lower latencies, however: at a 10 microsecond latency, light only travels 3,000 meters, providing sufficient time of flight resolution for a wide variety of applications, and sub-microsecond latencies are certainly possible[29].

Because the time per hash $\tau$ is set to $\tau = 5\times10^{-7}$ or 0.5µs/hash for Solana[25], network latency is likely to be the limiting factor in all practical proximity proofs rather than $VDF$ resolution.

## Existing network limits

Proximum's peer to peer PoP measurement does not map well to most existing physical network infrastructure.

Two physically proximate nodes may communicate through a distant hub, making PoP difficult due to the increased time of flight between the hub and the spokes, in addition to generally impractical traditional network latencies.[29] Rather than an obstacle, this may be a significant opportunity for Proximum.

Other networks such as Helium already enable point-to-point radio communication channels[14]. A similar approach can be used by Proximum nodes. Additionally, the speed parameter $c$ may be set to an artificially lower value such as 100,000,000$\frac{m}{s}$ and gradually increased as the network matures to account for initial network inefficiencies. Finally, new networks such as Starlink use a grid topology which may be able to resolve some of these latency concerns in addition to serving as future Proximum nodes[26]. Together, these forces serve as a gradual incentive for Proximum nodes to adopt or create low-latency decentralized mesh networks.

## 4 Structure

Proximum uses the H3 geospatial index and proof of proximity to create a tree of subnets. A small subset of these hierarchical H3 cells is illustrated in Figure 4. In this section we assume that every H3 resolution is used, but in practice restricting nodes to a subset of allowed H3 resolutions e.g. $\{0, 3, 7\}$ is likely to be more performant due to a reduced number of proofs required to reach consensus and more secure due to reaching a larger quorum of nodes in each subnet. The exact choice will be determined based on the result of the Security and Scalability simulation below.

### Subnet membership

The set of nodes within a subnet $N_n$ is defined by the set of nodes which can trustlessly verify their proximity to the majority of other nodes within $N_n$ contained in H3 cell $I_n$ with area-equivalent circle diameter $l_{I^r}$ through the n-way position estimation algorithm above.

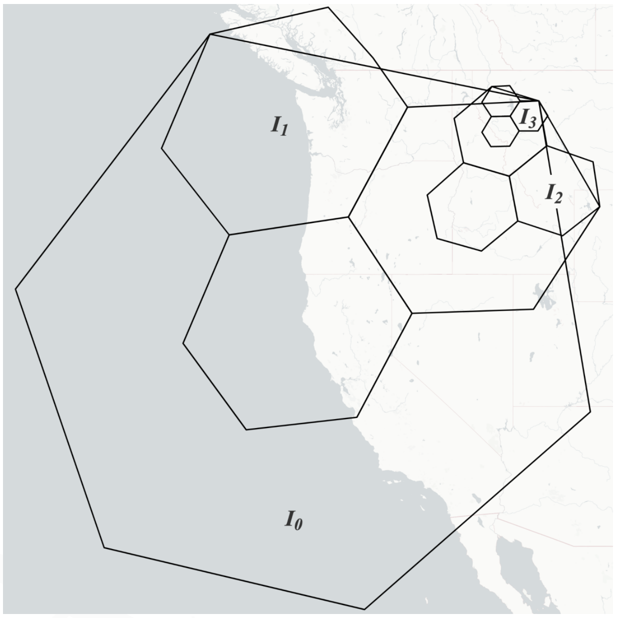

Figure 2: Selected H3 geospatial cells tiling the western United States at resolutions 0–3. Note the cell hierarchy: each cell $I_i$ is composed of 7 cells $I_{i+1}$, but cell borders do not line up precisely.

Each subnet $N_{I_r}$ is a validium to its parent subnet $N_{I_{r-1}}$. By validium, we mean that the computations performed on the subnet are verified by the parent subnet through a zero-knowledge proof, but the data required for these computations is not stored by the parent[6].

Once $b$ transactions have been verified within a subnet, they are "rolled up" into a single zero-knowledge proof and submitted to the parent subnet as a single transaction.

Network consensus is reached when the majority of $r_0$ subnets agree. The tree of subnets will be initially sparsely populated at network genesis: after launch, new subnets can either be created adjacent to an existing subnet or within an exsting subnet, preserving connectivity of the tree and the feasibility of high resolution proximity proofs between adjacent cells.

Nodes join a subnet $N_{I_r}$ by signing a mesage including the subnet's H3 cell $I_r$ and submitting it to the parent H3 cell $I_{r-1}$.

## 5 Consensus

Network consensus is reached when the majority of $r_0$ subnets agree on state, with voting power proportional the the approximate volume of spacetime swept by each node. Because all transactions on higher resolution subnets are included in this state, consensus at the network root implies consensus at all higher resolutions.

## 6 Computation

Proximum aims to preserve a general model of computation. Computation performed within a subnet must be defined by functions which are:

- Representable using the LLVM's intermediate representation (IR)[19] or multi-level intermediate representation (MLIR)[18],
- Compiled to a sandboxed target such as the extended Berkeley Packet Filter (eBPF)[5, 33] or WebAssembly (WASM)[31], and
- Provable by Proximum's zero-knowledge proof system.

This minimal set of restriction permits a broad variety of use cases such as peer-to-peer electronic cash transfers, trustless market-making, and machine-learning computation.

Note that Proximum nodes do not differentiate between "free" read operations and "paid" write operations: all function calls incur a cost proportional to their computational complexity.

## 7 State

The data upon which computation acts is represented as mapping from account to arbitrary bytecode. Each datum also includes several pieces of metadata: the set of subnets which must verify state transitions, the set of owners or accounts permitted to modify the state, and a deposit of the native asset ⚹.

address $\rightarrow$ (bytes, $\{I_n, I_m, \ldots\}$, {owners}, deposit)

Note that every datum within the state defines the set of subnets which must verify state transitions. This enables network users to select appropriate security parameters for each datum. The owners of the datum are also defined: these are the accounts which are permitted to modify the state.

### State propagation

(Add a section on the propagation of state consensus from local subnets to the entire network)

This deposit is debited by subnet nodes at a rate proportional to the length of the bytecode and the time elapsed since deposit. Once the balance reaches zero, the account is deleted.

### Examples

We can consider a few common examples: a smart contract can be deployed by saving the contract's bytecode to a datum. The contract can be upgradeable if the list of owners includes the address of another contract or user. An ERC20-like token could be implemented as a single datum containing token balances with a single owner: the implementation logic of the contract which defines the state-transition rules for the token's datum, e.g. for token mints and transfers.

### State migration

There may be cases where a datum's state is migrated to a new set of subnets in order to adapt to changing circumstances. The Proximum implementation should include a mechanism enabling this.

## 8 Economics

There are several economic considerations for Proximum. The native asset ⚹ is intended to be a money-like stable unit of account which is created, used, and burned within the Proximum network. Widely adopted networks successfully incentivize participants to perform the rewarded activity, e.g. Bitcoin's incentive model results in tremendous resources expended on SHA256 hashing[21]. Proximum incentivizes nodes to build a global, low-latency, peer-to-peer computational consensus network by rewarding nodes proportionally to the quantity of spacetime that they mine.

We discuss how each of the following economic parameters incentivizes this behavior in more detail below.

| Parameter | Value |
|---|---|
| Initial reserve | $p_{\text{init}} = 100,000,000 \ast$ |
| Mining rate | $p_{\text{mining}} = 100,000,000 \frac{\ast}{\text{year}}$ |
| Bond | $p_{\text{bonding}} = 100,000 \ast$ |
| Burn fraction* | $p_{\text{burn}} = 0.05$ |
| State rent rate† | $p_{\text{state}} = 1e-9 \frac{\ast}{\text{byte year subnet}}$ |
| Computation rate† | $p_{\text{comp}} = 1e-9 \frac{\ast}{\text{flop subnet}}$ |

Table 1: Economic Parameters. * denotes globally-adjustable parameters. † denotes subnet-specific adjustable parameters.

These internal use cases for $\ast$ are important because a good currency has "fallback" economic utility other than moneyness[15].

## Reserve

An initial supply $p_{\text{init}}$ is reserved at genesis to support network growth and initial node bonds.

## Bonding

Nodes must bond a quantity of $\ast$ to participate in consensus: $0 \leq \text{bond}_n \leq p_{\text{bond}}$ where $p_{\text{bond}_{\max}} = 100,000\ast$. Proximum is *not* a proof of stake network: consensus voting power is not proportional to the quantity of $\ast$ bonded. Instead, bonded $\ast$ provides economic incentive to follow Proximum algorithms correctly. Nodes which fail to follow the consensus algorithm may lose their entire bond.

## Mining

To incentivize nodes to fill spacetime, they earn newly minted $\ast$ for participating in consensus. Mining rewards are proportional to the spacetime volume asserted by each node. The target mining parameter $p_{\text{mining}} = 100,000,000 \ast$ per year. H3's authalic radius of 6,371.0072 km and Earth's approximate surface area $A_{\text{Earth}} = 510,065,616 km^2$[11] result in an areal mining rate of 0.196 $\ast$ per km$^2$ per year. Note that in practice the initial total mining rate will be lower than $p_{\text{mining}}$ because most of Earth's surface is inhospitable to mining nodes.

Instead of computing the complex spacetime surface swept by a mining node, e.g. by integrating its Voronai cell over time, mining rewards are scaled with the age of the bonded subnet assertion up to an age parameter $p_{\text{age}} = 1$ year and divided between all nodes within a single cell. The naive mining rate $\ast'_n$ for a node $n$ in cell $I_n^r$ at resolution $r$ is thus:

$$\ast'_n = \min\left(1, \frac{\text{age}}{p_{\text{age}}}\right) \frac{p_{\text{mining}}\text{bond}_n}{|n_N| p_{\text{bond}_{max}}} \frac{A_{I_n^r}}{A_{\text{Earth}}} \qquad \left(\frac{\ast}{\text{year}}\right)$$

where $A_{I_n^r}$ is the area of cell $I_n^r$ and $|n_N|$ is the number of nodes in cell $I$. Areas and average annual mining rates are shown for typical H3 resolutions in Table 2.

Note that if the Proximum network expands to locations beyond Earth, the mining rate $p_{\text{mining}}$ will stay constant and therefore the mining rate per existing subnet will decrease.

## Slashing

Nodes lose bonded $\ast$ to adversarial nodes for misbehavior, such as attesting to an invalid transaction or $VDF$ output. Any node that submits a proof of an invalid attestation by another node may claim the entire bond of the other node.

Nodes which fail to follow the consensus algorithm may lose their entire bond. This is a significant economic incentive to follow the consensus algorithm correctly.

| $r$ | $\|I^r\|$ | $A^r$ ($m^2$) | $l_{I^r}$ (m) | $\ast'_{I_n}$ $\frac{\ast}{\text{year}}$ |
|---|---|---|---|---|
| 0 | $1.2 \times 10^2$ | $4.3 \times 10^{12}$ | $2.3 \times 10^6$ | $8.3 \times 10^5$ |
| 1 | $8.4 \times 10^2$ | $6.1 \times 10^{11}$ | $8.8 \times 10^5$ | $1.2 \times 10^5$ |
| 2 | $5.9 \times 10^3$ | $8.7 \times 10^{10}$ | $3.3 \times 10^5$ | $1.7 \times 10^4$ |
| 3 | $4.1 \times 10^4$ | $1.2 \times 10^{10}$ | $1.3 \times 10^5$ | $2.4 \times 10^3$ |
| 4 | $2.9 \times 10^5$ | $1.8 \times 10^9$ | $4.7 \times 10^4$ | $3.5 \times 10^2$ |
| 5 | $2.0 \times 10^6$ | $2.5 \times 10^8$ | $1.8 \times 10^4$ | $5.0 \times 10^1$ |
| 6 | $1.4 \times 10^7$ | $3.6 \times 10^7$ | $6.8 \times 10^3$ | 7.1 |
| 7 | $9.9 \times 10^7$ | $5.2 \times 10^6$ | $2.6 \times 10^3$ | 1.0 |

Table 2: Statistics for H3 cells $I^r$ at resolution $r$, including cell count $|I^r|$, cell area $A^r$, area-equivalent circle diameter $l_{I^r}$, and maximum mining rate $r_m$

### State rent

Proximum users must pay an ongoing cost to maintain state on-chain. This parameter $p_{\text{state}}$ is intended to prevent long-term state bloat. If a state balance drops to zero, the nodes within that subnet may delete the state and withdraw the state rent. This dynamic parameter may be adjusted per subnet as storage costs and network usage patterns change.

### Computation

Proximum users must pay a one-time cost per computation. This dynamic parameter is adjusted per-subnet based on network load.

### Payments

Proximum users may transfer $\ast$ to other users.

### Burning

A fraction of all existing $\ast$ paid to nodes is burned rather than received by the node, specifically during bond slashing, state deletion, and computation payments. This global parameter $p_{\text{burn}}$ is intended to reduce long-term inflation and may be adjustable by the network.

## 9 Security

We must consider typical blockchain security considerations as well as new security considerations related to physical locality.

### Byzantine fault tolerance

A system is Byzantine fault tolerant if it has a solution to the Byzantine Generals Problem[17], which considers a distributed system with one or more nodes providing conflicting information. This can occur in blockchains due to faulty nodes or malicious nodes attempting to subvert the network.

In general a system reaching consensus by passing messages between nodes, $3m + 1$ nodes can tolerate at most $m$ Byzantine nodes[17]. Most proof of stake systems such as Ethereum and Solana are subject to this constraint.

Bitcoin exceeds this $3m+1$ threshold in practice because conflicting message sets can be independently ordered by quality (the total quantity of hashing work done on each message set[21]). The Bitcoin network can tolerate Byzantine nodes with $m$ hashpower as long as $> m$ honest nodes remain.

Does Proximum enable a similarly heightened level of Byzantine fault tolerance? The answer is not immediately clear. Conflicting sets of messages can be sorted by the quantity of spacetime they represent the and sum of bonds at risk, but both claims rely on economic incentives rather than physical constraint. An ideal proof of spacetime would be rivalrous such that no two nodes could physically claim the same region of spacetime at the same time: in this case nodes could always independently select the same message set with more confidence. Pending further analysis, we will assume that Proximum is subject to the same $3m + 1$ constraint as most other distributed systems.

### Forks

Nodes which attest to contradictory forks forfeit their entire bond on both forks. Any other node can submit each message to the alternate fork, and every fork can verify messages to be attestations for other forks within the same time period.

Typical bond slashing algorithms have limited effectiveness due to long range attacks, where a node bonds, withdraws, and then forks the network from a state prior to withdrawal to avoid losing their bond. Proximum avoids this concern due to a well defined $VDF$ clock which can be used to define a maximum fork time shorter than the bond withdrawal time.

## Double spends

A double spend occurs when a user submits two contradictory transactions are both accepted by the network (i.e. having one's cake and eating it too).

## Sybil attacks

A Sybil attack occurs when a single attacker falsely claims a large number of identities in order to influence a system where voting power is dependent on the number of identities[8]. In the case of Proximum, voting power is proportional to the volume of spacetime swept by each node, so a Sybil attacker would claim to be in many locations.

### Multiple keys, single location

Can an adversarial node attempt to increase its voting power by creating multiple nodes within a subnet? Yes, in the naive case where membership within a subnet is simply determined by completing a PoP to that subnet: all of an attacker's sybil copies will be able to perform the same proof.

Therefore we must modify this naive requirement to bound voting power in subnet $N_i$ per child subnet $N_{i+1}$ so that Sybil attacks result in at most 1/7 of the voting power of the parent subnet $N_i$. This solution could be continued to bound voting power in $N_i$ per child subnet $N_{i+2}$, restricting a sybil attacker to 1/49 of the voting power of the parent subnet $N_i$, and so on.

### Single key, multiple locations

The analysis above assumes that each node $n$ posseses a unique keypair $K_n$. In practice, a sybil attacker may share a key across multiple physical locations, but this is less of a concern. The attacker could demonstrate a lower proximity than expected based on the difference between its claimed and actual locations, which would allow it to participate in consensus for multiple subnets. But because the key *is* present in multiple subnets, this is a reasonable outcome.

The attacker could also offer a "location spoofing" service for users in which it signs transactions attesting to a user's presence in location A when they are actually in location B. There are two ways to mitigate this: network users should rely on subnet consensus on location rather than the attestation of a single node, and this fraud is detectable by honest nodes due to the time of flight constraint. Thus nodes signing unphysical proximity proofs can be caught and lose their bond.

### Multiple keys, multiple locations

This is no longer a sybil attack because the attacker is honestly representing their control of a large volume of spacetime. The network must instead rely on Byzantine fault tolerance to ensure the attacker cannot subvert it.

## Lazy nodes

Nodes which attest to incorrect signatures or VDF outputs may be caught by other nodes and lose their bond.

## Location collapse

The proof of proximity algorithm does not directly place a lower bound on the distance between nodes.

A pathological solution is placing a set of Proximum nodes within a small region, asserting a larger set of locations within a region containing insufficient honest nodes, and adjusting time-of-flight proofs to be consistent with the falsely asserted locations in order to fraudulently claim rewards, harm the honest nodes, or otherwise affect the network. This failure mode was frequently observed in the Helium network [1] and was addressed with a denylist of suspicious nodes [13]. Proximum may be able to create stronger guards against this attack by combining the following techniques:

### Initial conditions

The network can be seeded with a global set of nodes which are known to be physically distributed (e.g. through personal setup).

Given an intially correct setup with widely distributed nodes, new adversarial nodes will be unable to consistently deceive other nodes. Consider an adversarial node $n$ located at geospatial index $I_n$ but asserting a fraudulent geospatial index $I'_n$ separated by its real location by vector $\vec{\Delta I} = I'_n - I_n$. The adversarial node will be unable to prove its asserted proximity to nodes located in roughly the direction opposite $\vec{\Delta I}$ because the true distance will be greater than the claimed distance.

### Contiguous growth

New nodes may be permitted to join the network only within specified distance of existing subnets and within set rates. This prevents the creation of mostly-fraudulent node subnets which could overwhelm the signal from legitimate nodes.

### Stability

Once the Proximum network has a large number of globally distributed participants, asserting locations accurately is likely to be the most profitable strategy for any given node without particular knowledge of other fraudulent node locations, forming a Schelling point for all participants[22]. Note that all Proximum nodes and messages must avoid the Earth's interior. One can imagine an alternate network in space without this restriction. In that case, the Schelling point would likely be to misreport each node's location some (small but likely increasing) distance $\vec{\delta I}$ toward the centroid of the network. But because the Earth's interior excludes this strategy, there is no clear direction towards which all nodes could manipulate their locations without nodes on the opposite side of the Earth consistently failing proximity proofs.

### Clock synchronization

In practice nodes would find it difficult to maintain synchronized VDFs because $\tau$ varies per node based on clock speed and architecture. Attackers can go to extreme measures to decrease $\tau'$ such as using a liquid nitrogen cooled overclocked computer in order to exceed the VDF output rate of other nodes.

This attack can be avoided through a round-robin approach where additional randomness is fed into the VDF at set intervals, e.g. for each proof of proximity, upon each transaction receipt, or similar. This prevents an attacker from performing a long-range attack which allows them to generate $VDF$ outputs perceived by other nodes as "far in the future" and restricts them to less harmful short-range attacks such as misrepresenting their proximity by a factor of $\tau'/\tau$.

## 10 Scalability

We can simulate network scalability with the following assumptions:

- A message's starting subnet is uniformly drawn from all subnets at all used H3 resolutions.
- A message's destination subnet is likely to be nearby: there is linear falloff as distance from the source subnet increases.
- Only two cells are involved in each message

Subnet resolutions will be selected based on simulation results.

## 11 Applications

We consider several specific applications for Proximum, highlighting those which would not be feasible in prior blockchains.

### Truly local flatcoins

Local currencies have been used throughout history with varied success, even recently in the fiat era[4][3][9][16]. Proximum enables the creation of a truly local currency which only permits transactions within a defined spatial volume. Because the state and consensus set for this currency need only include subnets within this volume, the cost to transact with this currency on Proximum should be far smaller than an equivalent transaction on a blockchain requiring globally-defined consensus.

Consider a set of regional currencies: despite potentially disjoint consensus sets for each local currency, the Proximum consensus tree would still allow cross-regional trade from one local currency to another.

Currencies aiming to maintain stability over time and with respect to local economic conditions are termed flatcoins[27]. For local flatcoins implemented on Proximum, cross-regional trade would be visible as a net flow of value across the surface bounding the local economic region. This balance of trade measurement might be useful for inferring the relative economic strength of a region and trustlessly adjusting local currency parameters, i.e. creating an algorithmic local central bank to support stable price levels.

### High-frequency trading

Past blockchain networks have not been suitable for high frequency trading[32] because various methods require extremely low latencies (to the point that trading and exchange venues are often colocated to reduce delays due to the speed of light). Proximum's local subnets may enable high-frequency trading within specific subnets while preserving compabibility with the rest of the network.

### Location-gated transactions

Proximum enables the use of proximity as a new on-chain primitive. This could be used for allow-list transaction gating, such as check-ins for local merchant loyalty programs, on-chain geocache-style puzzles, or deny-list gating, such as geofencing specific regions in order to comply with local regulations.

### Jurisdictional nexus

The location of an activity is often relevant to legal or regulatory analysis and enforcement. Because Proximum provides a trust-minimized assertion of the location of transactions and state, it may be useful to establish that a particular activity *does* or *does not* occur within a particular jurisdiction.

## 12 Interoperability

Proximum can be used by other cryptographic networks.

### Location measurement

Networks unable to directly measure proximity of a user $u$ to a specific region can rely on Proximum as follows. Let $\{I_n, I_m, ...\}$ be a set of H3 cells covering the region. Any Proximum user $u$ can submit any transaction with a sufficiently low delay to a corresponding subnet $\{N_n, N_m, ...\}$ to demonstrate its proximity to that subnet. The separate network can then verify that transaction's inclusion in state consensus in several ways: reading an updated datum from a node within a subnet, checking the proof signatures against nodes known to occupy the subnet, or following the transaction's proofs $\Pi$ to the root subnet to ensure the transaction was included in global consensus.

### Consensus adoption

An existing network may adopt Proximum's proof-of-spacetime consensus mechanism by following these steps:

1. Formulating its state transition logic for compatibility with Proximum
2. Reaching agreement with a set of subnets to adopt this logic
3. Adding proximum nodes to the existing network rules and legacy consensus mechanism
4. Disabling the legacy consensus mechanism

Because only selected subnets are required to adopt the third party network logic, and because the prior state of the third party network can be treated as an intrinsic for those nodes, migration to Proximum's consensus model should be less expensive than most potential chain migrations.

## 13 Future work

There are several areas where further research might yield useful results.

### Data availability

Because each subnet is a validium to its parent, data needed to reconstruct subnet state is not stored by the parent. This is the parsimonious choice: consensus is an $O(n)$ problem while data storage is an $O(1)$ problem. But it also requires trust: one either must store the relevant data oneself or trust another party to store it. Proximum's modularity allows nodes to use third parties to perform that service, but building data availability into Proximum might increase its utility: computation and data storage are naturally complementary services.

### Scalability

There are likely many opportunities to improve transaction and proof speeds by optimizing network topology, message format, data format, and transaction format. Because the data involved in each transaction is enumerated, the potential for parallelization within a node is high. This paper is agnostic on such details in order to leave room for these optimizations.

### Space-based locations

Because the goal for Proximum's consensus algorithm is maximizing decentralization and security, placing nodes in space is a natural extension of the network. This expansion might offer several advantages: it could incentivize the development of commercial proof-of-spacetime satellites, possibly marking the first profitable space-based mining; enhance network resilience against terrestrial threats, including adversarial nation-state actions; and establish a legal foundation for provably extra-terrestrial applications. However, H3 is unsuitable for orbital dynamics, necessitating an additional solar location indexing system. Because space-based nodes could claim far larger regions of spacetime than terrestrial nodes at a far lower economic value density, the compensation per volume of spacetime might need to be reduced for these nodes.

## 14 Conclusion

Proximum is a computational consensus protocol including a tree of local subnets, nodes with voting power proportional to asserted spacetime volume, a trust-minimized proof of proximity, and an economic incentive structure for network participants using a native asset $\ast$. This network enables new classes of application which require trust-minimized location measurement.

# Appendix

## Byzantine fault tolerance

TBD

## One-way hash functions

A one-way hash function takes an input message or pre-image $m$ and converts it to a fixed length hash value $h = H(m)$ where $h$ serves as a hard-to-forge fingerprint of $m$. Proximum uses the SHA3 family of hash functions, specifically SHA-256[24].

Intuitively, hashing the message creates a unique, hard-to-forge fingerprint of that message. Specifically we assume a hash function with the following properties exists [23, 24]: * Pre-image resistance: Given a hash value $h$ it is infeasible to find a pre-image $M$ such that $h = H(M)$. * Second pre-image resistance: Given a pre-image $M_1$ it is infeasible to find a distinct message $M_2$ that results in the same hash value $h = H(M_1) = H(M_2)$. * Collision resistance: it is infeasible to find any distinct pre-images $M_1$ and $M_2$ which result in the same hash value $h = H(M_1) = H(M_2)$. Note that any collision-resistant hash function is also second pre-image resistant.

## Verifiable delay functions

A Verifiable Delay Function (VDF) provides 1: a minimum bound on elapsed time from receiving a value $x$ to finding a value $y$ and an optional proof $\pi$ by performing $n$ sequential iterations of an evaluation function, and 2: a simpler method to verify this calculation.[2]

$$(y, \pi) = V^n_{\text{Eval}}(x)$$

$$V_{\text{Verify}}(x, y, n, \pi) = \begin{cases} \text{True}, & \text{if } (y, \pi) = V^n_{\text{Eval}}(x); \\ \text{False}, & \text{otherwise.} \end{cases}$$

For an effective VDF, $V^n_{\text{Eval}}$ has time complexity $O(n)$, and we permit $V_{\text{Verify}}(x, y, n, \pi)$ to also have time complexity $O(n)$ with a sufficiently speed improvement ($> 1000$), despite the original authors' requirement that $V_{\text{Verify}}(x, y, n, \pi)$ have a time complexity of $O(\text{polylog}(t))$.[2] Proximum uses a sequential SHA-256 $V_{\text{Eval}}$ and a parallel SHA-256 $V_{\text{Verify}}$ based on [33] for simplicity:

$$\begin{aligned} (y, \pi) &= V^n_{\text{Eval}}(x) = H^n(x) \\ \pi_0 &= x \\ \pi_i &= H^{n/m}(\pi_{i-1}) \\ \pi_m &= y \end{aligned}$$

where $H^2(x)$ is defined as $H(H(x))$, $H^n(x)$ represents the $n$-fold composition of $H$ with itself, and $\pi$ splits $n$ into $m$ equal parts.

The verification function achieves a linear speedup by calculating each of the proof's $m$ intermediate hash values in parallel:

$$V^n_{\text{Verify}}(x, y, n, \pi) = \begin{cases} \text{True}, & \text{if } \forall i > 0, \\ & \quad \pi_i = H^{n/m}(\pi_{i-1}); \\ \text{False}, & \text{otherwise.} \end{cases}$$

Elapsed time $\Delta t$ may be calculated based on the time $\tau$ required per SHA256 hash and the number of iterations $n$:

$$\begin{aligned} \Delta t &= \tau n, \\ \tau &= 5 \times 10^{-7} \text{ seconds/hash.} \quad [25] \end{aligned}$$

Proximum uses this VDF as a trustless clock and trustless pseudo-random value generator: we assume that all Proximum nodes begin with a synchronized $x_0$ at $t_0$ and can thus agree on the ordering and timing of events: $t_i = \tau i + t_0$ is the time at which $y_i = V^i_{Eval}(x_0)$ is calculated by a Proximum node.

## Digital Signatures

A keypair $K_n$, consisting of a public key $K_{n_{\text{pub}}}$ and a private key $K_{n_{\text{priv}}}$, can be used to sign an arbitrary message $m$ using the signature algorithm $S$. Any observer can use the corresponding verification algorithm $R$ to verify that a signature $s$ was produced by keypair $K_n$ for message $m$.

$$s = S_{K_{n_{\text{priv}}}}(m),$$
$$R_{K_{n_{\text{pub}}}}(m, s) = \begin{cases} \text{True}, & \text{if } s \text{ is a valid signature} \\ & \text{for } m \text{ by } K_n; \\ \text{False}, & \text{otherwise.} \end{cases}$$

Let $M(x_1, x_2, \dots)$ be an invertible function for serializing structured data $x$ into a message $m$, such as Google's Protocol Buffers[10]. The serialization function $M$ can be expressed as:

$$m = M(x).$$

For convenience, we denote a signed message that contains both the internal message data $x$ and a signature $s$ of that data using keypair $K_n$ as $M_{S_n}(x)$:

$$= M(x, S_{K_{n_{\text{priv}}}}(x)).$$

Any observer can inspect a signed message and verify that the signature matches the keypair and message contents. Proximum nodes use digital signatures, specifically the Ed25519 algorithm, to attest to specific messages[7].

## Zero-knowledge proofs

Proximum relies on zero-knowledge proofs to ensure computation has been performed as specified. The proof system has not been selected yet because the author is relatively unfamiliar with the field and because the tradeoff between proof complexity and interactivity may mean that selecting an interactive but simpler proof system is a reasonable tradeoff for Proximum given the requirement for low-latency peer-to-peer communication already present in the proof of proximity.

## Spatial indexing

The H3 geospatial index tiles the Earth's surface into hexagonal cells at distinct resolutions, denoted as $r$, as illustrated by Figure 4. These 16 hierarchical resolutions range from very large ($r = 0$: 122 cells, each roughly 2,000 km wide) to very small ($r = 15$: 5.7e14 cells, each about 0.8 meters wide). H3 also defines a mapping from an integer index $I_r$ to a specific cell and resolution $r$, providing a convenient integer representation for both location and location precision.[12, 11]

Note the bitmask used to identify the resolution and cell index of a location $I$ at resolution $r$ from Figure 4 above:
$I_0 =$ `8029fffffffffff`,
$I_1 =$ `8128fffffffffff`,
$I_2 =$ `822897fffffffff`,
$I_3 =$ `83289afffffffff`.

Proximum node $n$ can sign a message $M_{S_n}(I_r, \dots)$ to assert its location within a specific cell $I_r$, and Proximum defines the subnet $N_{I_r}$ as the set of nodes asserting a location within cell $I_r$.

## Speed of light

Information cannot be be transmitted faster than the speed of light, denoted as $c$:

$$c = 299,792,458\,\text{m/s} \quad [30].$$

Proximum uses the speed of light to prove proximity of a keypair $K$ based on the observation that light travels about 150 meters during the time period $\tau$ required to compute one iteration of $V_{\text{Eval}}$.

## Extended Kalman Filter

(Summary of standard EKF principles)

## References

[1] 3Roam. *Helium Hotspot Spoofing and the Deny List.* `https://3roam.com/helium-hotspot-spoofing-and-the-deny-list/`. Accessed: 2024-01-27. 2022.

[2] Dan Boneh et al. *Verifiable Delay Functions.* https://eprint.iacr.org/2018/601.pdf. 2018.

[3] *Berkeley Bucks.* https://www.visitberkeley.com/things-to-do/berkeley-bucks/. Accessed: 2024-02-05.

[4] *BerkShares - Local Currency for the Berkshire Region.* https://berkshares.org/. Accessed: 2024-02-05.

[5] *BPF Documentation - Linux Kernel.* https://www.kernel.org/doc/html/latest/bpf/index.html. Accessed: 2023-10-05.

[6] Vitalik Buterin. *Understanding Layer 2 Scaling Solutions.* https://vitalik.eth.limo/general/2023/10/31/l2types.html. Accessed: 2024-01-29. Oct. 2023.

[7] *Digital Signature Standard (DSS).* Tech. rep. National Institute of Standards and Technology, 2019.

[8] John R Douceur. *The Sybil Attack.* Tech. rep. Accessed: 2024-01-29. IPTPS, 2002.

[9] Jonathan Goodwin. *Free Money Miracle.* https://mises.org/library/free-money-miracle. Accessed: 2024-02-05.

[10] Google. *Protocol Buffers.* https://developers.google.com/protocol-buffers. Accessed: 2023-01-27. 2023.

[11] *H3 Core Library Resolution Table.* https://h3geo.org/docs/core-library/restable. Accessed: 2024-01-25.

[12] *H3 GitHub Repository.* https://github.com/uber/h3. Accessed: 2024-01-25.

[13] Helium. *Denylist.* https://github.com/helium/denylist. Accessed: 2024-01-27. 2022.

[14] *Helium Network.* https://www.helium.com/. Accessed: 2024-01-25.

[15] Guido Hülsmann. *The Ethics of Money Production.* Ludwig von Mises Institute, 2008. ISBN: 1933550090. URL: https://www.amazon.com/Ethics-Money-Production-Guido-H%C3%BClsmann/dp/1933550090.

[16] *Ithaca Hours.* https://en.wikipedia.org/wiki/Ithaca_Hours. Accessed: 2024-02-05. 2023.

[17] Leslie Lamport, Robert Shostak, and Marshall Pease. "The Byzantine Generals Problem". In: *ACM Transactions on Programming Languages and Systems* (July 1982), pp. 382–401. URL: https://www.microsoft.com/en-us/research/publication/byzantine-generals-problem/.

[18] Chris Lattner et al. "MLIR: Scaling Compiler Infrastructure for Domain Specific Computation". In: *2021 IEEE/ACM International Symposium on Code Generation and Optimization (CGO).* 2021, pp. 2–14. DOI: 10.1109/CGO51591.2021.9370308.

[19] LLVM Project. *LLVM Language Reference Manual.* https://llvm.org/docs/LangRef.html. Accessed: 2024-02-06. 2024.

[20] B. B. Mandelbrot. *The Fractal Geometry of Nature.* W. H. Freeman and Co., 1982.

[21] Satoshi Nakamoto. *Bitcoin: A Peer-to-Peer Electronic Cash System.* https://bitcoin.org/bitcoin.pdf. 2008.

[22] Thomas C. Schelling. *The Strategy of Conflict: With a New Preface by the Author.* Harvard University Press, 1981. ISBN: 9780674840317. URL: https://www.hup.harvard.edu/books/9780674840317.

[23] Bruce Schneier. *Applied Cryptography.* https://www.schneier.com/books/applied-cryptography/. Accessed: 2024-01-25.

[24] *SHA-3 Standard: Permutation-Based Hash and Extendable-Output Functions.* Tech. rep. National Institute of Standards and Technology, 2015.

[25] *Solana Clock Ticks Per Second.* https://github.com/solana-labs/solana/blob/ef233eaaa7aa20fbdae870fc82e37bd26f0f885d/sdk/program/src/clock.rs. Accessed: 2024-01-25.

[26] Calum Spring-Turner and Raj Rajan. *Performance Bounds for Cooperative Localisation in the Starlink Network*. July 2022. DOI: 10.48550/arXiv.2207.04691.

[27] Balaji Srinivasan. *Tweet on Flatcoin*. Twitter. https://twitter.com/balajis/status/1402030233319088139. June 2021.

[28] Paul Storcz. "Security Budget in the Long Run". In: *Truthcoin.Info* (2024). URL: https://www.truthcoin.info/blog/security-budget/.

[29] *The Role of Latency in Online Gaming*. https://escholarship.org/uc/item/5d79r601. Accessed: 2024-01-25.

[30] *Value of the Speed of Light*. https://physics.nist.gov/cgi-bin/cuu/Value?c. Accessed: 2024-01-25.

[31] *WebAssembly Core Specification*. https://webassembly.github.io/spec/core/. Accessed: 2023-10-05.

[32] Wikipedia. *High-frequency trading*. https://en.wikipedia.org/wiki/High-frequency_trading. Accessed: 2024-02-05. 2023.

[33] Anatoly Yakovenko. *Solana: A new architecture for a high performance blockchain*. Tech. rep. Available online. Solana, 2017. URL: https://solana.com/solana-whitepaper.pdf.