# Helium

## A Decentralized Machine Network

Amir Haleem Andrew Allen Andrew Thompson Marc Nijdam Rahul Garg

Helium Systems, Inc.

Release 0.3.1 (2018-05-09)

## Abstract

The Internet of Things is an $800 billion industry, with over 8.4 billion connected devices online, and spending predicted to reach nearly $1.4 trillion by 2021 [1]. Most of these devices need to connect to the Internet to function. However, current solutions such as cellular, WiFi, and Bluetooth are suboptimal: they are too expensive, too power hungry, or too limited in range.

Helium is a *decentralized machine network* that enables machines anywhere in the world to wirelessly connect to the Internet and geolocate themselves without the need for power-hungry satellite location hardware or expensive cellular plans. Powering the network is a blockchain with a native protocol token incentivizing a two-sided marketplace between coverage providers and coverage consumers. With the introduction of a blockchain, Helium injects decentralization into an industry currently controlled by monopolies. The result is that wireless network coverage becomes a commodity, fueled by competition, available anywhere in the world, at a fraction of current costs.

Helium's secure and open-source primitives enable developers to build low-power, Internet-connected machines quickly and cost-effectively. Helium has a wide variety of applications across industries and is the first decentralized machine network of its kind.

## 1. Introduction

The world is becoming decentralized. A multitude of platforms, technologies, and services are moving from centralized proprietary systems to decentralized, open ones. Peer-to-peer networks such as Napster (created by Helium founder Shawn Fanning) [2] and BitTorrent paved the way for blockchain networks and crypto-currencies to be built. Now Bitcoin, Ethereum, and other blockchain networks have shown the value of decentralized transaction ledgers. Existing Internet services such as file storage, identity verification, and the domain name system are being replaced by modern blockchain-based versions. While software-level decentralization has moved quickly, physical networks are taking longer to affect. These networks are more complicated to decentralize as they often require specialized hardware to function.

Helium is a wide-area wireless networking system, a blockchain, and a protocol token. The blockchain runs on a new consensus protocol and a new kind of proof, called *Proof-of-Coverage*. Miners who are providing wireless network coverage in a cryptographically verified physical location and time submit proofs to the network, and the miners submitting the best proofs are elected to an asynchronous byzantine fault tolerant consensus group at a fixed epoch. The members of the consensus group receive encrypted transactions submitted by other miners and forms them into blocks at an extremely high transaction rate. In addition to the blockchain protocol, the Helium Wireless protocol, *WHIP*, provides a bi-directional data transfer system between wireless machines and the Internet via a network of independent providers that does not rely on a single coordinator, where: (1) machines pay to send & receive data to the Internet and geolocate themselves, (2) miners earn tokens for providing network coverage, and (3) miners earn fees from transactions, and for validating the integrity of the network.

**Note:** This whitepaper represents a continuous work in progress. We will endeavor to keep this document current with the latest development progress. As a result of the ongoing and iterative nature of our development process, the resulting code and implementation is likely to differ from what is represented in this paper.

We invite the interested reader to peruse the Helium GitHub repo at `https://github.com/helium` as we continue to open-source various components of the system over time.

### 1.1 Key Components

Helium is built around the following key components:

***Proof-of-Coverage*** We present a computationally inexpensive *Proof-of-Coverage* that allows miners to prove they are providing wireless network coverage. We anchor these proofs using a *Proof-of-Serialization* that allows miners to prove they are accurately representing time relative to others on the network in a cryptographically secure way.

**Blockchain Network** We demonstrate an entirely new purpose-built blockchain network built to service the *Wireless Protocol* and provide a system for authenticating and identifying machines, providing cryptographic guarantees of data transmission and authenticity, offer transaction primitives designed around the wireless protocol, and more.

**Helium Consensus Protocol** We present a novel consensus protocol construction that creates a permissionless, high throughput, censor-resistant system by combining an asynchronous Byzantine Fault Tolerant protocol with identities presented via *Proof-of-Coverage*.

**Wireless Protocol** We introduce a new open-source and standards-compliant wireless network protocol, called *WHIP*, designed for low power machines over vast areas. This protocol is designed to run on existing commodity radio chips available from dozens of manufacturers with no proprietary technologies or modulation schemes required.

***Proof-of-Location*** We outline a system for interpreting the physical *geolocation* of a machine using WHIP without the need for expensive and power-hungry satellite location hardware. Machines can make immutable, secure, and verifiable claims about their location at a given moment in time which is recorded in the blockchain.

### 1.2 System Overview

- Helium is a *Decentralized Machine Network* built around a new wireless protocol (WHIP) on a purpose-built blockchain with a native token.
- Machines take the form of hardware containing a radio chip and firmware compatible with WHIP, and spend tokens by paying Miners to send data to and from the Internet.
- Miners earn tokens by providing wireless network coverage via purpose-built hardware which provides a bridge between WHIP and Routers, which are Internet applications.
- Machines store their private keys in commodity key-storage hardware and their public keys in the blockchain.
- Miners join the network by asserting their satellite-derived location, a special type of transaction in the blockchain, and staking a token deposit.
- Miners specify the price they are willing to accept for data transport and *Proof-of-Location* services, and Routers specify the price they are willing to pay for their Machine's data. Miners are paid once they prove they have delivered data to the Machines' specified Router.
- Miners participate in the creation of new blocks in the blockchain by being elected to an asynchronous Byzantine Fault Tolerant consensus group.
- Miners are rewarded with newly minted Helium for blocks that are created while they are part of the consensus group.
- A Miner's probability of being elected to the consensus group at a given epoch is based on the quality of the wireless network coverage they provide.
- The blockchain employs *Proof-of-Coverage* to guarantee that miners are honestly representing the wireless network coverage they are creating.

[Figure 1] shows a visual representation of the Helium network.

## 2. The Helium DMN

We introduce the Helium Decentralized Machine Network (`DMN`). The `DMN` provides wireless access to the Internet for machines by way of multiple independent miners, and outlines a network and wireless protocol specification by which participants in the network should conform. Routers pay this network of miners for sending data to and from the Internet, and miners are rewarded with newly minted tokens for providing network coverage and delivering machine data to the Internet.

### 2.1 Participants

There are three types of participants in the network: Machine, Miner, or a Router.

**Machines** send and receive encrypted data from the Internet using hardware compatible with the Helium wireless protocol, called *WHIP* [Section 2.4]. Data sent from machines is *fingerprinted*, and that fingerprint stored in the blockchain.

**Miners** provide wireless network coverage to the network via purpose-built hardware, called gateways [Section 2.5], which provide a long-range bridge between WHIP machines and the Internet. Users join the network as miners by purchasing or building a gateway that conforms to the wireless protocol, and *staking* a token deposit proportional to the density of other miners operating in their area [Section 5.3.3]. Miners participate in the *Proof-of-Coverage* [Section 3] process to prove that they are continuously providing wireless network coverage that machines can use. Miners join the network with a score [Section 3.3.4] that diminishes as blocks pass without valid proofs being submitted. At a given epoch, a new group of miners are elected to a *consensus group* which mine new blocks in the blockchain and receive the block reward and transaction fees for any transactions included in the block once mined.

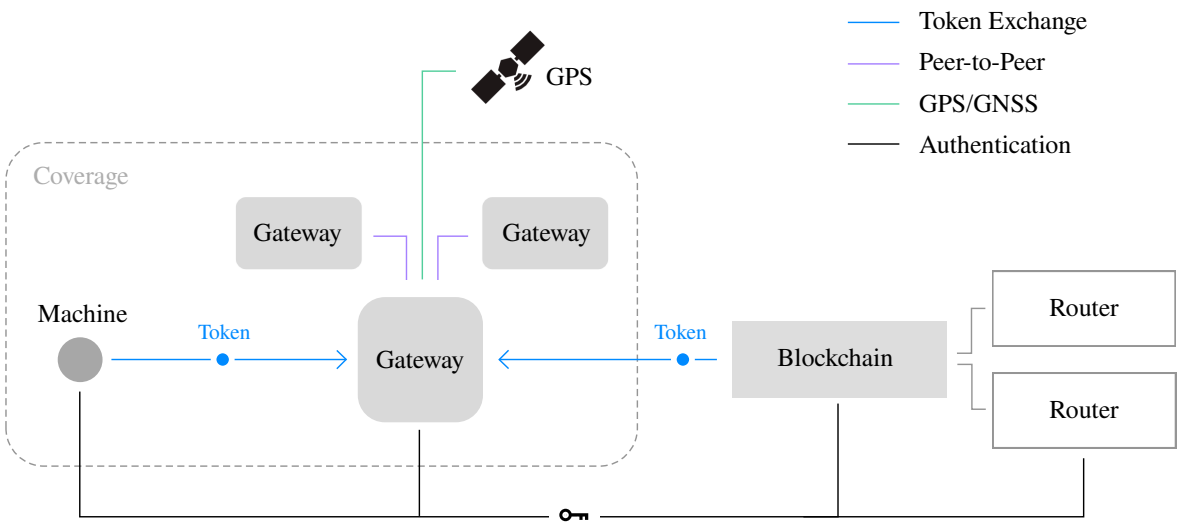


**Figure 1.** *System Overview*

As a miners score drops their probability of being elected to the consensus group and mining blocks diminishes.

**Routers** are Internet applications that purchase encrypted machine data from miners. In locations with a sufficient number of miners, routers can pay several miners to obtain enough copies of a packet to geolocate a machine without needing satellite location hardware, which we call *Proof-of-Location*. Routers are the termination point for machine data encryption. Machines record to the blockchain which routers miners should send their data to, such that any gateway on the network can send any machines data to the appropriate router. Routers are responsible for confirming to gateways that machine data was delivered to the correct destination and that the miner should be paid for their service.

### 2.2 Blockchain

The Helium blockchain is a distributed ledger designed to provide a cost-effective way to run application logic core to the operation of a `DMN`, store immutable machine data fingerprints, and furnish a transaction system. The Helium blockchain is an immutable append-only list of transactions which achieves consensus using the *Helium Consensus Protocol* [Section 6]. Users internal and external to the `DMN` have access to the blockchain, which is a new protocol built from scratch specifically for the `DMN`.

The blockchain consists of blocks which contain a header and a list of transactions. There are several kinds of transactions, outlined in [Section 5].

At a given epoch a given block consists of:

| Block Version |
|---|
| Block Height |
| Previous Block Hash |
| Transactions *1..n* Merkle Hash |
| Threshold signature by the current consensus group |

As the *Proof-of-Coverage* [Section 3] is valuable to the network, miners are required to submit their proofs at regular intervals. All miners have a score which decays over time, and is boosted by submitting *Proofs-of-Coverage* to the blockchain. At a fixed epoch, a *HoneyBadgerBFT* [4] consensus group of the highest scoring miners is elected. For that epoch, all transactions are encrypted and submitted to the consensus group for inclusion in the blockchain. The consensus group is responsible for decrypting transactions using *threshold decryption*, agreeing on the validity and ordering of transactions, forming them into blocks, and appending them to the blockchain for which the members of the consensus group receive a reward.

As the consensus group is validating transactions without having to provide an associated block-proof (beyond a threshold signature), there is practically no settlement time, and the transaction throughput is extremely high compared to a *Nakamoto Consensus* blockchain such as Bitcoin or Ethereum. The Helium Consensus Protocol is outlined in detail in [Section 6].

### 2.3 Physical Implementation

In addition to the blockchain network, Helium is a *physical* wireless network instantiation. The participants in the wireless network can be thought of as follows:

**Wireless Protocol** The wireless network uses a new open wireless protocol, called *WHIP*. WHIP is a long-range, low-power, wireless network protocol suitable for use with commodity open-standards hardware. WHIP compatible hardware can communicate over many square miles in dense urban environments or hundreds of square miles in rural settings. WHIP compatible hardware can also last for several years using standard batteries. WHIP uses strong public key cryptography and authentication occurs using the Helium blockchain, and data is encrypted end-to-end between the machine and corresponding Internet-hosted router.

**Gateways** Gateways are physical network devices that provide wide-area wireless coverage and participate in the blockchain network. Gateways transmit data back and forth between routers on the Internet and machines while generating *Proofs-of-Coverage* for the blockchain network [Section 3]. Gateways are manufactured using commodity open-standards components with no proprietary hardware. Gateways can co-operate and geolocate machines using the wireless network without any additional required hardware. Each Gateway can support thousands of connected machines, and provide coverage over many square miles. Miners operating gateways specify the price they are willing to accept for transport and *Proof-of-Location* services for machines.

**Machines** Machines exist in the form of hardware products that contain a WHIP-compatible radio transceiver and communicate with gateways on the wireless network. WHIP is designed to facilitate low power data transmission and reception, so typically machines exist in the form of battery-powered sensors that can operate for several years using standard batteries (although mains-powered machines also work quite well). Machines can exist in a variety of forms, depending on the product or use case, and a variety of transmission and reception strategies can be employed to optimize for transmission/reception frequency or battery life. Machine manufacturers are encouraged to use hardware-based key storage which can securely generate, store, and authenticate public/private key pairs without leaking the private key.

In this section, we expand on the components of the wireless network.

### 2.4 Wireless Protocol (*WHIP*)

#### 2.4.1 Motivation

Several Low Power Wide Area Network (LPWAN) technologies are available today. These wireless technologies focus on creating long-range, low-power Internet communication for sensors and other smart machines. Typically these technologies trade throughput for range, with data rates as low as 18 bits per second (bps) and range measured in miles. In comparison, a typical WiFi network has significantly higher data rates but ranges limited to only a few dozen feet. Several of these new technologies, such as LoRa [6] and RPMA [7], have gained good traction and there are many commercial products available compatible with these systems. However, we believe a decentralized machine network should use non-proprietary protocols and modulation schemes and that participants in this ecosystem should have the freedom to choose between competing hardware vendors. We do not consider an open alliance built on top of proprietary hardware to be an acceptable compromise. While there are many open-standard wireless networking stacks, such as IEEE 802.15.4 [8] used in the first generation of Helium wireless products, none meet our extremely long range and low power criteria. It is this lack of open solutions that drove the creation of a new protocol.

#### 2.4.2 Outline

We introduce a new open-source Low Power Wide Area Network (LPWAN) protocol called *WHIP*. WHIP is a highly secure, long range, low power, bi-directional wireless network protocol that is compatible with a wide range of existing radio transceivers operating in the sub-GHz unlicensed frequency spectrum. Authentication with the wireless network uses modern public-key encryption and NIST P-256 ECC key pairs, with the public keys for all participants stored in the blockchain.

The modulation format is simple and widely supported, easy to implement and has excellent resistance to RF noise. There are dozens of vendors implementing radio transceivers compatible with WHIP, such as Texas Instruments, Microchip, and Silicon Labs.

WHIP is a *narrowband* wireless protocol which creates several channels within the unlicensed spectrum and employs frequency hopping to switch between channels. Typically frequency hopping requires a complex time-synchronized system that is limited in capacity. However, machines using WHIP do not need to coordinate with gateways on channel selection as gateways are capable of hearing all channels within the available spectrum at any time. We choose narrowband to accomplish the following goals:

**Spectral Efficiency** It is necessary to operate within unlicensed RF spectrum very efficiently. RF is a shared, lim-

ited resource, and therefore a focus on efficiency to increase capacity and improve robustness is a must.

**Co-Existence Performance** As the number of machines and networks increase, the ability to operate in *noisy* RF environments without interference is a critical consideration.

**Range** Narrowband allows for extremely long-range communications, with data rates that scale both up and down depending on the density of gateways.

#### 2.4.3 Implementation

WHIP supports several data rates, channel bandwidths, and error-correction techniques. Gateways and machines dynamically negotiate the combination of these options using a *signalling packet* delivered at the lowest bandwidth and symbol rate to ensure maximum range for the initial communication.

The full WHIP specification will be made available by the Decentralized Machine Network Alliance in the coming months.

### 2.5 Gateways

Gateways are physical network machines operated by miners that create wireless RF coverage over wide areas. They transmit data back and forth between routers on the Internet and machines on the network, process blockchain transactions, and create *Proofs-of-Coverage* for the blockchain network [Section 3]. Gateways can connect to the Internet using any TCP/IP capable backhaul, such as Ethernet, WiFi or Cellular. Each gateway contains a radio frontend chip capable of listening to several MHz of radio spectrum at a time and can hear all network traffic transmitted within that spectrum. In this configuration modulation and demodulation is done in software, which is typically referred to as a *Software Defined Radio (SDR)*. The benefit of this structure is that gateways can hear any machine traffic transmitted within the frequency range, and no synchronization between the gateway and machine needs to occur. This allows machines to remain inexpensive and relatively simple and reduces wireless protocol overhead. If a miner wishes to minimize their gateway hardware costs, synchronized frequency hopping schemes are also permitted within the specification as a cheaper alternative to a more expensive radio frontend.

Gateways require a GPS or GNSS receiver to obtain accurate position and date/time information. This satellite-derived location is used in conjunction with other techniques to verify that a gateway is, in fact, providing wireless network coverage in the location it claims. Because satellite location messages are easy to fabricate and do not necessarily prove that wireless RF coverage is being created, multiple mechanisms are required to validate this work as described in more detail in [Section 3].

Satellite location information is also correlated with packet arrival events to provide *Proof-of-Location* for machines if multiple gateways observe the same packet. This allows machines to locate themselves without requiring a GPS/GNSS transceiver physically, and therefore provide accurate location data at a fraction of the battery life and cost of competing methods. This method is described in detail in [Section 4].

Helium Systems Inc. will make both a complete open-source reference design and a finished product available at network launch.

### 2.6 Machines

A *machine* is any wireless hardware capable of communicating with gateways via WHIP. WHIP is designed to facilitate low power data transmission and reception, so typically machines would exist in the form of battery-powered sensors that can function for several years using standard batteries.

WHIP is designed such that machines can be manufactured using commodity hardware available from a wide variety of vendors with a very low-cost bill of materials (BOM). The technology in modern radio transceivers, such as the Texas Instruments CC1125 or STMicroelectronics S2-LP, enables exceptionally long-range network systems that can be built without the need for proprietary modulation schemes or physical layers. Some of these radios are available for around $1 at reasonable volumes.

It is recommended that each machine use the Microchip ECC508A or equivalent hardware-based key storage device, which can securely generate, store, and authenticate public/private *NIST P-256 ECC* [3] key pairs without leaking the private key. Also, a wide array of defense mechanisms prevent logical attacks on the encrypted data between the key storage device and its host machine, along with physical protections on the security device itself. Users program their key storage device as part of the onboarding process defined in the WHIP wireless specification using a defined API.

### 2.7 Routers

Routers are Internet-deployed applications that receive packets from machines via gateways and route them to appropriate destinations such as an HTTP or MQTT endpoint. Routers also act as *full nodes* for the blockchain network [Section 5.5].

Routers serve several functions on the network, including:

- Authenticating machines with the wireless network
- Receiving packets from gateways and routing them to the Internet
- Delivering downlink messages, including OTA updates, to machines via gateways

- Providing delivery confirmations to ensure transport transactions are honest
- Providing authentication and routing mechanisms to third-party cloud services such as *Google Cloud Platform* or *Microsoft Azure*
- Storing and making available a full copy of the blockchain ledger

When a gateway receives a data packet from a machine on the network, it queries the blockchain to determine which router to use given the machine's network address. Anyone is free to host their own router and define their machines' traffic to be delivered there by any gateway on the network. This ability allows users of the network to create VPN-like functionality whereby encrypted data is delivered only to a router (or set of routers) that they specify and can optionally host themselves.

Routers can implement a system called a *Channel* which handles the authentication and routing of data to a specific third party Internet application, such as *Google Cloud Platform IoT Core*. These channel implementations can take advantage of a machine's onboard hardware security to create a secure, hardware-authenticated connection to a third party which would otherwise be difficult to implement directly on an embedded microcontroller. Helium will make available an open source reference implementation of a Channel that can be used to build additional interfaces to Internet services.

Helium Systems Inc will also host a high-availability cloud router for anyone to use and also provides and maintains an open-source router that is available either as source code or a binary package for a variety of operating systems and distributions.

The protocol specification required for implementing a router is defined in the WHIP Wireless Specification document that will be made available by the Decentralized Machine Network Alliance in the coming months.

## 3. *Proof-of-Coverage & Proof-of-Serialization*

In the Helium network, miners must prove that they are providing wireless network coverage that machines are able to use to communicate with the Internet. Miners do this by complying with the *Proof-of-Coverage* protocol which the blockchain network and other miners audit and verify. We use a *Proof-of-Serialization* to ensure that miners are correctly representing their time in relation to others on the network, and obtain cryptographic proof of dishonest behavior. Several components of the Helium network, such as *Proof-of-Coverage*, use *Proof-of-Serialization* as a cryptographic "anchor" that root those occurrences with a cryptographic time proof. With a combination of *Proof-of-Coverage* and *Proof-of-Serialization* we can obtain cryptographic proof of the approximate location and time of events occurring within the Helium network.

In this section we outline the motivation and implementation for *Proof-of-Coverage* and *Proof-of-Serialization*.

### 3.1 Motivation

Most existing blockchain networks such as Bitcoin [9] and Ethereum [5] use a *Proof-of-Work* system that relies on an algorithmic puzzle that is asymmetric in nature. These proofs are extremely difficult to generate, but simple for a third party to verify. Security on these networks is achieved by the network-wide *consensus* that the amount of computing power required to generate a valid proof is difficult to forge, and as subsequent blocks are added, the cumulative difficulty of the chain becoming prohibitively difficult to fabricate.

These computation-heavy proofs are, however, not otherwise *useful* to the network. We define useful as work that is valuable to the network beyond securing the blockchain. While there have been attempts in other networks to turn mining power into something useful, such as Ethereum executing small programs called *smart contracts*, the majority of the work is not useful or reusable. The mining process is also extremely wasteful, as the determining factor in the work is typically computational power which consumes massive amounts of electricity and requires significant hardware to execute.

The proofs used in Helium must be resistant to *Sybil Attacks* in which dishonest miners create pseudonymous identities and use them to subvert the network and gain access to block rewards to which they should not be entitled. This is a particularly difficult attack vector to manage in a physical network like Helium. We must also be resistant to a new attack vector: *Alternate Reality Attacks* exist where a dishonest group of miners are able to simulate that wireless network coverage exists in the physical world when it in fact does not. An example of this would be running the mining software on a single computer and simulating GPS coordinates and RF networking.

We later propose a consensus protocol [Section 6] that uses *Proof-of-Coverage* to both secure the blockchain and provide an extremely useful service to the network; providing wireless network coverage that machines can use to send data to and from the Internet.

### 3.2 Inspiration

*Proof-of-Coverage* (`PoC`) is an innovative proof which allows miners to prove that they are providing wireless network coverage $W$ in a specific region to a challenger, $C$. `PoC` is an interactive protocol where a set of targets $T_n$ assert that $W$ exists in a specific GPS location $L$ and then convinces $C$ that $T_n$ are in fact creating $W$ and that said coverage must have

been created using the wireless RF network. PoC is the first such protocol that attempts to prove the veracity of miners in a physical space, and then use it to achieve consensus on a blockchain network.

With PoC we aim to solve for the following:

- Our goal is to prove that miners are operating radio frequency (RF) hardware and firmware compatible with the wireless protocol
- Our goal is to prove that miners are located in the geography they claim by having them communicate via RF
- Our goal is to correctly identify which version of reality is correct when there is a conflict

*Proof-of-Coverage* is inspired by the *Guided Tour Protocol (GTP)* [13] which devises a system for denial of service prevention by requiring a client $c$ to make a request to a variety of "tour guide" computers $G_n$ in order to gain access to a server $s$. The tour guides must be visited in a specific order and a hash of data exchanged which reveals the location of the next $G_n$ in order. Only after every $G_n$ has been visited can $c$ gain access to $s$.

Once $c$ gets to the last stop of the tour, it submits evidence of the first and last stop to $s$ who is able to verify that the first and last stops of the tour are correct without needing to contact $G_n$, and that $c$ could only know the first and last stops if it had completed the tour correctly.

While an extremely clever and innovative system, GTP is not directly suitable as a proof in our wireless network as radio frequency (RF) networking has limited range and therefore cannot communicate with peers anywhere on the network. We aim to construct a proof loosely based on the ideas presented in GTP, but applicable to our protocol.

We combine *Proof-of-Coverage* with *Proof-of-Serialization*—a proof that allows miners on the network to achieve cryptographic time consensus among decentralized clients. We aim to achieve rough time synchronization in a secure way that does not depend on any particular time server, and in such a way that, if a time server does misbehave, clients end up with cryptographic proof of that behavior.

### 3.3 Constructing *Proof-of-Coverage*

This section describes the construction of the *Proof-of-Coverage* protocol.

We aim to construct a proof that takes advantage of the following characteristics of radio frequency (RF) communication that are unique and different to Internet communication:

1. RF has limited physical propagation and therefore distance
2. The strength of a received RF signal is inversely proportional to the square of the distance from the transmitter
3. RF travels at the speed of light with (effectively) no latency

Our goal is to verify whether miners in a physical region are acting honestly and creating wireless network coverage compatible with the Helium wireless protocol (WHIP). To do this, a challenger $C$ deterministically constructs a multi-layer data packet $O$ which begins at an initial target, $T_1$, and is broadcast wirelessly to a set of sequential targets, $T_n$, each of which are only able to decrypt the outer-most layer of $O$ if they were the intended recipient. Each target signs a receipt, $K_s$, delivers it to $C$, removes their layer of $O$, and broadcasts it for the next target. Essentially an "envelope of envelopes" only decipherable by the intended recipient.

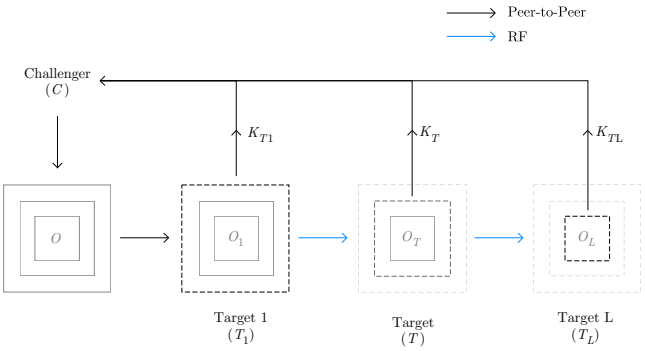


**Figure 2.** *Multi-Layer Data Packet Deconstruction*

#### 3.3.1 Selecting the Initial Target

We aim to deterministically locate a geographic reference target, $T$, for the challenger, $C$. Both $C$ and $T$ are miners in the network. $T$ does not need to be geographically proximate to $C$. To do so, $C$ initially seeds verifiable entropy, $\eta$, into the selection process by signing the current block hash with their private key. Since the probabilities associated to each miner form a discrete probability distribution [Equation 1], $C$ uses the probability associated to each eligible miner to locate $T$ and applies the inverse cumulative distribution function using a uniform random number generated via $\eta$. This allows us to ensure that we always target potentially dishonest miners as they have a lower score, thus increasing their probability of being targeted by $C$. Given that a miners score is diminishing linearly over time [Section 3.3.4], it is necessary to create this inverse relationship to give low-scoring miners an opportunity to participate in the process and increase their score. This diminishing score also incentivizes all the participants to send receipts to the challenger and broadcast the remainder of $O$.

#### 3.3.2 Constructing the multi-layer challenge

Once $T$ has been selected, $C$ must construct a multi-layer challenge, $O$. $O$ is a data packet broadcast over the wireless

network and received by geographically proximate targets $T_n$. Geographically proximate is defined as within a radius of $T$, a network value $T_{\text{radius}}$. Each layer of $O$, $O_l$, consists of a three-tuple of $E\left(S, \psi, R\right)$, where $E$ is a secure encryption function using the ECDH derived symmetric key, $S$ is a nonce, $\psi$ is the time to broadcast the next layer of the challenge and $R$ is the remainder of $O$ consisting of recursive three-tuples. The maximum number of $O_l$ is bounded by a network value, $O_{\text{max}}$.

The construction logic of $O$ by $C$ is as follows:

1. A set of candidate nodes, $T_n$, are selected such that all members of $T_n$ are within a contiguous radio network that also contains $T$
2. Two targets, $T_1$ and $T_L$, are selected by finding the highest scoring targets in $T_n$ furthest from $T$
3. A weighted graph, $T_g$, is constructed from $T_n$ such that members of $T_g$ in radio range of each other are connected by an edge weighted by the value of $1 - \Big(score(T_a) - score(T_b)\Big)$
4. The shortest path between $T_1$ to $T$ to $T_L$ is computed using Dijkstra's algorithm [10] using the edge weights from the previous step
5. An ephemeral public/private keypair $E_k$ and $E_{\text{k-1}}$ are generated
6. A layer $O_l$ is created and added to $O$, and $S$ is encrypted with the combination of the public key of $T_L$, retrieved from the blockchain as $T_{Lk}$ and $E_{\text{k-1}}$ as an Elliptic-Curve Diffie-Hellman (ECDH) exchange to compute a shared secret, known only to both parties $C$ and $T_L$
7. The previous step repeats with additional layers added to $O$ until all $T_L \rightarrow T_1$ have a layer $O_l$ included in $O$

The resulting $O$ can be visually represented as depicted in [Figure 3].

### 3.3.3 Creating the Proof

Once $O$ has been constructed, it is delivered to $T_1$ via the Internet peer-to-peer network and immediately broadcast by $T_1$ via the wireless network. The Helium wireless protocol is not a point-to-point system, so several miners within proximity of $T_1$ will hear $O$. As described prior, each layer $O_l$ of $O$ contains the three-tuple $E\left(S, \psi, R\right)$ where $E$ is a secure encryption function using the ECDH-derived key, $S$ is a nonce, $\psi$ is a time at which to transmit the next layer of the packet and $R$ is the remainder of $O$ consisting of recursive three-tuples. In this example, only the specific target $T$ will be able to decrypt $E$ and send a valid receipt back to the challenger, $C$.

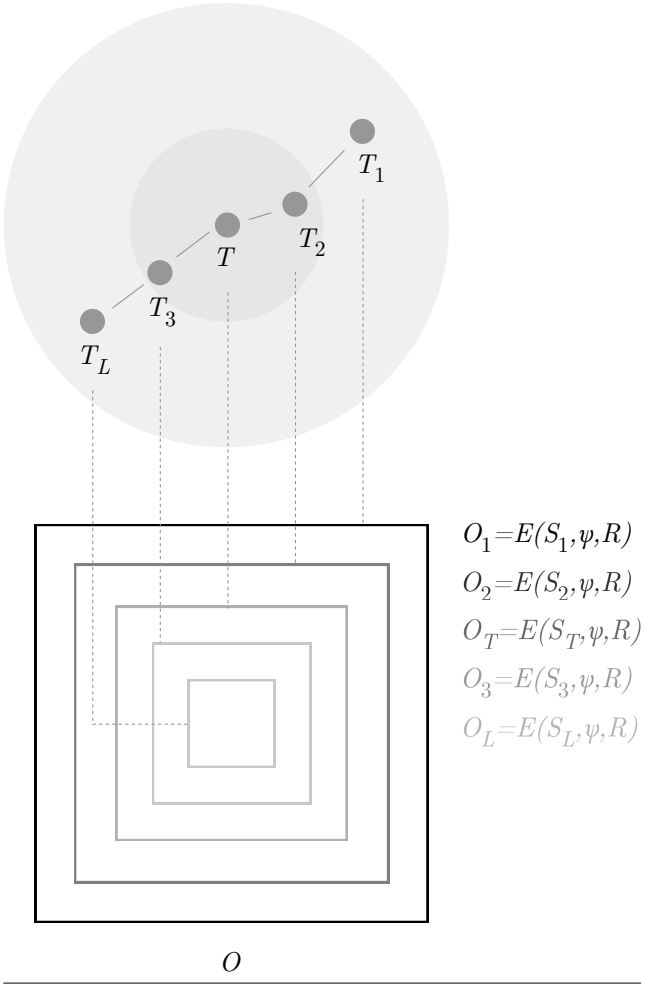


**Figure 3.** Construction of $O$

We describe the approximate flow of *Proof-of-Coverage* creation as follows:

1. $T_1$ receives $O$ from $C$ via the peer-to-peer Internet network, decrypts the outermost layer and immediately broadcasts it $R$ via the wireless network
2. $T$ hears $O$ and attempts to decrypt the value of $E$ by using its private key $pk$ : $E_{pk}\left(S, \psi, R\right)$
3. $T$ records both the time of arrival $\beta$ and the signal strength $\upsilon$ of $O$
4. If successful, $T$ then creates signed receipt $K_s$, $K_s = (S||\beta||\upsilon)$ signed by the private key of $T$
5. $T$ submits $K_s$ to $C$ via the peer-to-peer Internet network, removes the outer most layer, and wirelessly broadcasts the remainder $O$
6. These steps repeat for $T_1..T..T_L$, with $T_L$ being the last target in the graph

$C$ expects to hear responses from $T_g$ within a time threshold $\lambda$, otherwise it considers the *Proof-of-Coverage* to have concluded. Because $C$ is the only party with complete knowledge of $O$, upper bounds of the values for $\beta$ and $\upsilon$ are assigned by $C$ which are used to verify that each layer of $O$ was transmitted approximately where and when it was expected. The upper bound for $\beta$ ls limited by the speed of light $\tau$ between $T_n$ and $T_n - 1$. Thus we know that, subject to some slight delays from reflection or multipath, the packet should not arrive at $T_g$ later than $\tau$ multiplied by the geographical distance $D$ plus some small episilon value, $\upsilon = \tau \times \Big(D + \epsilon\Big)$. For $\upsilon$, because of the inverse-square law, we can calculate the maximum RSSI (Received Signal Strength Indication) possible for a packet transmitted, $\mu$, from $T_g - 1$ to $T_g$ as $\mu = \frac{1}{D^2}$. Gateways that are closer than expected, or which are transmitting at a higher power to mask their location disparity, are unlikely to get $\mu$ correct, given that they do not know who the next layer of $O$ is addressed to.

Once $T_L$ has delivered receipt to $C$, or $\lambda$ has elapsed, the *Proof-of-Coverage* is completed. The collection of signed receipts, $K_s$, constitute the *Proof-of-Coverage* that $C$ will submit to the network.

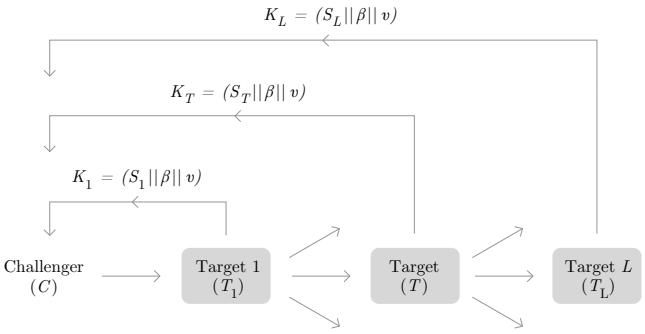


**Figure 4.** *Proof-of-Coverage* flow

### 3.3.4 Scoring

The score allocated to a miner, and therefore the resulting score of the *Proof-of-Coverage*, is an integral part of the consensus protocol, further outlined in [Section 6]. When miners join the Helium network, they are assigned a score, $\phi_m$. We consider any miner with a score greater than $\phi_m$ to be an *honest miner*. This score depreciates according to the number of verifications the miner has as well as the height since their last successful verfication. As $\phi_m$ decreases the probability of the miner $M$ being the target for $C$ increases such that the network continually attempts to prove that the lowest scoring miners are acting honestly, and giving miners a reasonable chance to improve their score.

In order to achieve this behavior we define the following invariants:

$M$, miner
$\upsilon$, number of successful verfications for $M$ - number of failed verifications for $M$
$h$, height since the last successful verification for $M$

If we assume that the ideal verification interval for any miner is close to 240 blocks (4 hours if we assume a 60 second block time), we scale these invariants to fit the scoring functions:

$\upsilon'$, $\upsilon/10.0$
$h'$, $h/480$

Using the above we can now construct a staleness-factor, $\delta$, which would be used in determining the score of the miner $M$.

$$\delta m = \begin{cases} -(8.h')^2 & \upsilon' = 0 \\ \upsilon'.(1 - \frac{h'^2}{min(0.25,\upsilon')}) & \upsilon' > 0 \\ \upsilon'.(1 - 10.\upsilon'.h'^2) & \upsilon' < 0 \end{cases}$$

The above conditions strictly adhere to the following principles:

1. A negative $\upsilon$ indicates that the miner is consistently failing verification.
2. If $\upsilon$ = 0, we don't have any trust information, therefore, we use a steep parabolic curve for the decay dependent on $h'$.
3. If $\upsilon > 0$, it implies that the miner has been successfully verified consistently, hence, we use an inverse parabolic curve that crosses the Y axis at 1, where the width of the parabola increases as a factor of $\upsilon$ up to 0.25. This implies that the more positive verifications [Section 3] the miner has accrued, the slower its score decays as a factor of $h'$.
4. Finally, if $\upsilon < 0$, this is the inverse of the above case, wherein, a miner has consistently been failing verification. Therefore, we use a similar parabola as above, however, the width of the parabola decreases as a factor of $\upsilon$, leading to a higher score decay for the miner as a factor of $h'$.

[Figure 5] shows the trends for each of the above functions.

Adhering to the above set of rules, we define the following scoring function, which is essentially a variation of a sigmoid curve fluctuating between values (0, 1):

$$\phi_m = \frac{\arctan(2.\delta m) + 1.58}{3.16}$$

This scoring function yields [Figure 6], which shows the variation of the score with the staleness-factor:

[Figure 7] shows a snapshot of a random subset of the network at any blockchain height $h$. The miners represent random locations with an illustrated score, while the edges are calculated using Dijkstra's algorithm [10].

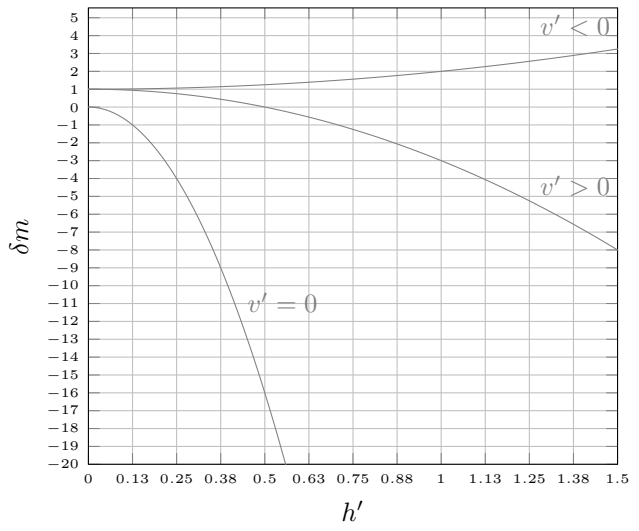


**Figure 5.** *Trendlines for the scoring functions*

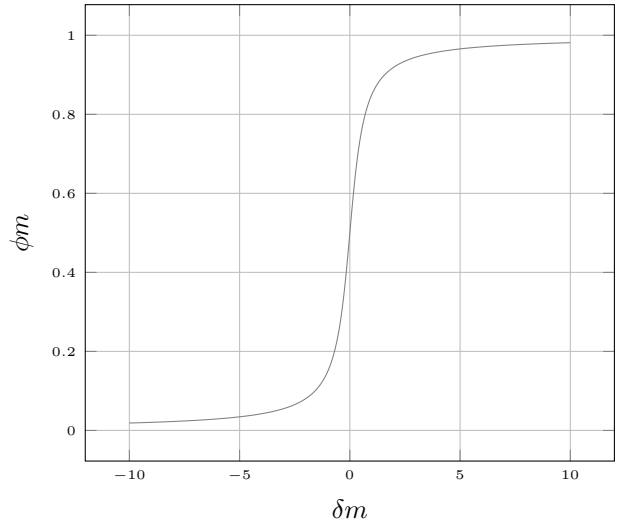


**Figure 6.** *Scoring algorithm and the resulting staleness factor*

After 10,000 iterations the network appears as represented in [Figure 8].

The goal of this system is to ensure that the scoring algorithm considers that some miners may attempt to act dishonestly. However, because the calculated edge-weights (via Dijkstra's algorithm) and the target selection mechanism ensure that we only boost the score of a miner when it is being verified by other high scoring miners, we believe that the system will favor legitimate miners and deter dishonest ones.

### 3.3.5 Target Selection

Due to the way scoring decays, there is a possibility that a given miners' score may become stale as that miner may not

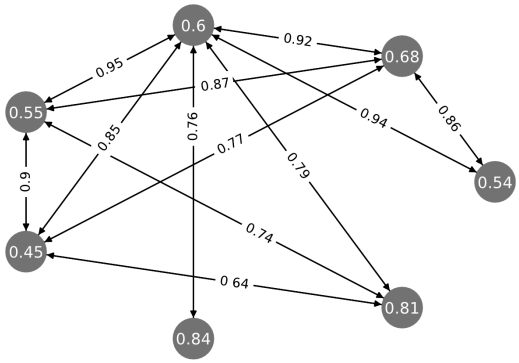


**Figure 7.** *Snapshot of a random subset of the initial network*

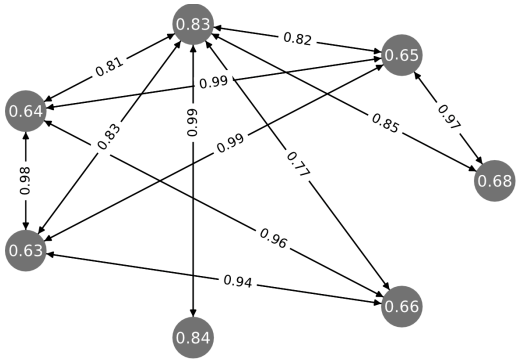


**Figure 8.** *Snapshot of a random subset of the network after 10000 iterations*

be verified within a reasonable interval. We therefore structure the target selection mechanism to give miners a statistically greater chance to increase their score by being selected as a target as their score decays. This is accomplished by biasing the probability of miners being selected as potential targets based on their individual scores.

Let the set of miners be defined as:

$$N = \{m_1, m_2, m_3 \dots \ m_n \mid n > 1\}$$

Let the set of miner scores be defined as:

$$S = \{\phi_m, m \in N\}$$

We assign the target selection probability to each miner in the following way:

$$P(m) = \frac{1 - \phi_m}{n - \sum_{i=1}^{n} \phi_{m_i}} \tag{1}$$

The above equation ensures that the miner with the lowest score is assigned the highest probability of being selected as a potential target while the opposite holds for the miner with the highest score.

Furthermore, it also asserts that the probabilities are inversely proportional to the score of an individual miner. This allows us to successfully target potentially low scoring miners and improve the overall balance of the scoring system.

Another valuable aspect of assigning the probability as shown above is that all the probabilities together form a discrete probability distribution. A discrete probability distribution satisfies the following equation:

$$\sum_{i} P(M = i) = 1$$

#### 3.3.6 Verifying the Proof

Once $T_L$ has delivered $K_s$, or $\lambda$ has elapsed, the *Proof-of-Coverage* is considered complete. When $C$ submits this proof, via a special type of transaction, all receipts $K_s$ from $T_1...T_L$ are included in the transaction published to the blockchain network. As all the steps originally completed by $C$ are deterministic in nature with verifiable and recreatable randomness, it is simple for a verifying miner, $V$, to recreate the original steps and verify that the proof is legitimate.

Verifying miners in the consensus group [Section 6] who see the proof transaction are able to verify the *Proof-of-Coverage* by recreating the following steps:

1. The verifying miner, $V$, reconstructs the set of miners $N$
2. The random seed $\eta$ can be verified by $V$ to have been created at approximately the correct time by the private key of $C$
3. $V$ then selects $T$ from $N$, as seeding with $\eta$ will result in the same target selection
4. The set of candidate $T_n$ are reconstructed from which $T_1$ and $T_L$ are determined
5. Dijkstra's algorithm is used to reconstruct the graph $T_g$
6. The $K_s$ receipts contained in $B_C$ are verified to have been signed by the private keys of $T_1..T..T_L$

Assuming these steps are completed successfully, the *Proof-of-Coverage* is verified the score of $C$ is adjusted appropriately.

### 3.4 Constructing *Proof-of-Serialization*

To achieve cryptographic time consensus among decentralized clients, we implement a simplified form of Google's Roughtime [12]. Roughtime is a protocol that aims to achieve rough time synchronization in a secure way that does not depend on any particular time server, and in such a way that, if a time server does misbehave, clients end up with cryptographic proof of that behavior.

This section describes the construction of the *Proof-of-Serialization* protocol.

#### 3.4.1 Creating the Proof

We outline the approximate process to achieve cryptographically secure time as follows:

1. To begin, a miner $M$ pseudo-randomly picks two miners $M_1$ and $M_2$, to prove contact serialization with
2. It is assumed $M$ has a public key for $M_1$ and $M_2$, otherwise $M$ should obtain it from the blockchain
3. $M$ generates a nonce, $R$, which is a SHA512 hash of the *Proof-of-Coverage* which $M$ has partially constructed
4. $M$ then generates a salted hash commitment $K$ called the *proof-kernel* $K = H\left(R||M_1||M_2\right)$
5. $M$ sends $K$ to $M_1$. $M_1$ replies with $T$, a signed message including the current time $T_1$ and $K$
6. $M$ knows that the reply from $M_1$ was not pre-generated because it includes the nonce $R$ that $M$ generated

Because $M$ can not trust $M_1$ it will ask for another time from $M_2$:

1. For the second request, a new nonce $R$ is generated using $T$ truncated to 512-bits, blinded by XOR'ing a randomly generated 512-bit number
2. $M$ then generates a sub-proof-kernel, $L = H\left(R||T||K\right)$, and sends it to $M_2$
3. $M_2$ replies with $U$, a signed message including the current time $T_2$ and $L$
4. $U$ is now a proof artifact that shows that $M$ desired and then proved a serialization between $M_1$ and $M_2$

With only two servers, $M$ can end up with proof that something is wrong, but no idea what the correct time is. But with half a dozen or more independent servers, $M$ will end up with chain of proof of any server's misbehaviour, signed by several others, and enough accurate replies to establish what the correct time is, $T_t$.

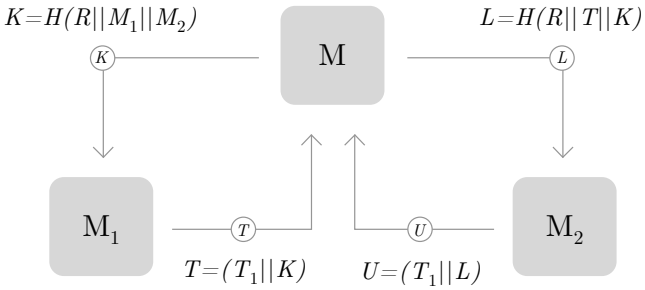


**Figure 9.** *Creating Proof-of-Serialization*

### 3.4.2 Verifying the Proof

If we assume that the times from $M_1$ and $M_2$ are significantly different, and the time from $M_2$ is before $M_1$, then $M$ has proof of misbehaviour. The reply from $M_2$ implicitly shows that it was created later because of the way that $M$ constructed the nonce. If the time from $M_2$ is after, then $M$ can reverse the roles of $M_1$ and $M_2$ and repeat the process to obtain, assuming steady clocks, a misordered proof as in the other case.

To verify the correct time, it is necessary for $M$ to repeat the time synchronization process with enough miners to gain consensus on the correct time:

1. A miner $M$ again pseudo-randomly selects $n$ miners $M_1 ... M_n$
2. $M$ generates a salted hash commitment $K$ and delivers it to $M_1$, where $K = H\left(R||M_1||M_2\right)$
3. $M_1$ again responds with $T$, a signed message containing the current time $T_1$ and $K$
4. $M$ generates a sub-proof-kernel, $L = H\left(R||T||K\right)$, and sends it to the next miner $M_n$
5. The next miner replies with $U$, a signed message including the current time and $L$
6. These steps repeat through $M_n$ until at least three time responses, $T_n$, are monotonic
7. $T_n$ can then be confirmed to be $T_t$, the correct time

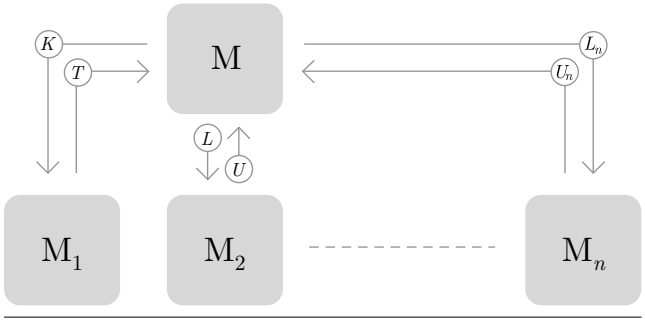


**Figure 10.** *Verifying Proof-of-Serialization*

### 3.4.3 Utilizing the Proven Time

Once the correct time $T_t$ has been determined via *Proof-of-Serialization*, it is used by $M$ and included during proof construction as described in [Section 2.2]. The randomness $\eta$ used to compute $O$ and thus obtain the *Proof-of-Coverage* is tied to the previous block, which contains $T_t$. This allows us to prove, with relative certainty, that some piece of data $D$ was created between the time of the previous block $b_t$ and $T_t$. $D$ in this case is the *Proof-of-Coverage*. Thus we know that $D$ must have been constructed between $b_t$ and $T_t$. This ensures that the *Proof-of-Coverage* cannot be pre-computed.

## 4. *Proof-of-Location*

Using *Proof-of-Coverage* and *Proof-of-Serialization* we achieve cryptographic proof of a miners location and cryptographic time consensus among those miners; we can take advantage of these proofs to determine the physical *geolocation* of WHIP-compatible machines and generate a new type of proof based on the machines geolocation. We call this *Proof-of-Location*.

### 4.1 Motivation

Location tracking is one of the most valuable use cases for low power machines. It is expected that there will be at least 70 million asset tracking devices shipping by 2022 [14].

Today, Global Navigation Satellite Systems (GNSS) are used by the majority of machines which require geolocation services, with GPS being the most popular implementation. GPS systems use a technique called *Time of Arrival (TOA)* to determine the location of a receiver in relation to 20 or so satellites orbiting the earth. GPS satellites synchronize their time using a high precision on-board clock and regular synchronization with control servers on the ground. GPS receivers receive precisely timestamped data from a number of satellites overhead and use a technique called *trilateration* to provide a precise location on earth.

GPS has matured into an extrordinarily reliable service used in a wide range of applications for providing both location and time services. However, there are significant drawbacks to GPS particularly in the realm of low power machines that Helium is designed to facilitate. It can take around 2 minutes for a GPS receiver to achiehve *lock* with sufficient satellites, which translates to drastically reduced battery life. As an example, a machine transmitting its location around 25 times a day may only last a month on a AA battery compared with several years of life on the same battery without GPS. Using GPS indoors is generally impossible, as the GPS receiver typically needs line of sight with the sky in order to see the 3-4 satellites required to calculate an accurate location. GPS data is also delivered unencrypted, which leaves the system extremely vulnerable to spoofing, jamming, and other attack vectors.

We are interested in low power implementations of location services that, in conjunction with an immutable distributed ledger, can be used to verify location and time. Given the above factors, we conclude that GPS is an unacceptable mechanism for these requirements.

This section outlines a new geolocation implementation that we describe as *Proof-of-Location*: a low power, cryptographically provable immutable record of a machines location and time.

### 4.2 Constructing *Proof-of-Location*

Our goal is to verify the physical geolocation of a given machine, $D$, without using GNSS hardware. To do this, we rely on the fact that we have already determined and proven the physical geolocation and cryptographic time consensus of a given miner, $M$, using the *Proof-of-Coverage* and *Proof-of-Serialization* protocols described in [Section 3].

#### 4.2.1 Precise timestamping of RF data

There are a handful of techniques used by positioning systems without the use of GNSS, which include Received Signal Strength Indication (RSSI), Time of Arrival (ToA), and Time Differential of Arrival (TDoA). These techniques use radio frequency transmissions, usually recieved by one or more receivers, combined with various algorithms based on characterstics of those transmissions.

Our conclusion is that TDoA is the most accurate but challenging technique to implement [15], [16], [17], [18]. TDoA, in simple terms, relies on the variance between precisely synchronized and recorded timing information between a transmitter and several receivers. As such it is critical to accurately *timestamp* RF packets machines emit, and synchronize the clocks of miners on the network.

An example timestamping flow is as follows:

1. A machine, $D$, broadcasts a packet $P$ containing arbitrary data via the wireless network

2. Several miners, $M_n$, hear $P$, and record a timestamp $T_n$ of their reception time of $P$

3. $T_n$ is created based on the nanosecond time received via GNSS and stamped using raw radio sample data received by the gateway radio frontend

4. A signed transaction including $P$ and $T_n$ are delivered to the router $R$ belonging to $D$ by $M_n$

5. $R$ has now received several copies of $P$, each of which has a slightly varying value of $T_n$

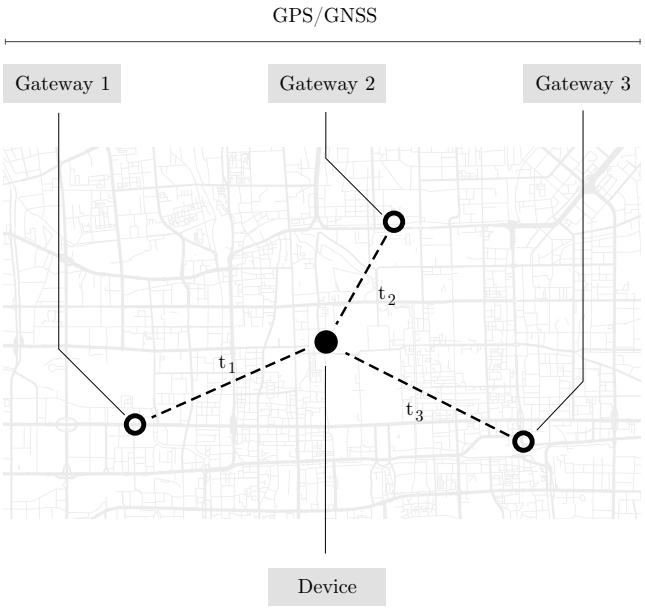


**Figure 11.** *Geolocation via TDoA*

Typically it is challenging to accurately record these timestamps as any nanosecond-level variance in the timestamp can lead to significant variance in the resulting location solution. To achieve this level of precision it is necessary to use extremely high-bandwidth raw *in-phase and quadrature (I/Q)* data from the miners radio hardware and a fast enough processor to sample this data, identify an appropriate packet, and record the timestamp. Typically a *Field Programmable Gate Arrays (FPGA)* is used as the processor for this data as these types of processors are able to process data in a deterministic way. However FPGAs are fairly expensive, power hungry, and emit significant heat. Instead, the reference Helium Gateway mining hardware uses a novel technique using commodity low-cost components to process I/Q data and achieve timestamping at this level of precision. As a comparative example, an existing low-cost LoRaWAN [23] access point is only capable of providing timestamp data accurate within several milliseconds of precision - as radio waves travel at the speed of light, each millisecond equates to approximately 300,000 meters of physical distance which we deem practically useless for any accurate geolocation. Further information on the techniques, components and schematics used in the reference Helium Gateway will be released as open source software at network launch.

#### 4.2.2 Using timestamps to derive location

Now that the machines router, $R$, is in posession of a variety of signed messages which include the precise timestamps $T_n$ it is possible to *solve* for the location of the machine $D$. A variety of TDoA algorithms exist such as [20], [21], [19] and [22]. If a sufficient density of $M_n$ and therefore $T_n$ are recorded for a given packet, the location of $D$ can be derived

down to a few meters depending on a variety of factors. We encourage the interested reader to read the cited papers for further details on TDoA algorithms, as they are beyond the scope of this document.

#### 4.2.3 Verifying *Proof-of-Location*

Once $R$ has computed a location of $D$, it may become necessary to verify that the reported location of $D$ was at accurate at that given moment in time. As the *Proof-of-Location* is deterministic and derived from information publicly available in the blockchain it is possible to reconstruct every step involved:

- From the signatures contained within the timestamped packets, $T_n$, every miner involved in providing timestamps can be verified
- By inspecting the `assert_location` [Section 5.3.3] transaction, the claimed GPS location of those miners can be determined
- The *Proofs-of-Coverage* and scores [Section 3] for each miner can be retrieved from the blockchain and inspected

By auditing the above steps the router operator can cryptographically prove (or disprove) the location of each of the miners involved in providing the components for *Proof-of-Location* for a given machine $D$.

The accuracy of the proof will depend heavily on the number of $M_n$ involved and therefore $T_n$ received. Additional RF factors such as reflections and multipath can significantly affect the accuracy of the location calculation.

## 5. Transactions

Transactions in the Helium blockchain provide functionality that enables address-to-address transfers of Helium tokens, similar to many existing blockchain networks, but also provide a set of primitives that enable core functionality that is critical to the operation of a `DMN`. In this section we address the philosophy behind our transaction system, the primitives, and the way fees on the network work. We will first address Helium's need for microtransactions and propose a new solution.

### 5.1 Helium's Need for Microtransactions

**Machines Pay Per Packet** The goal of Helium is to provide Internet data transport fees (the fees paid by machines to miners) that are an order of magnitude less than anything currently available for this type of service. This transport fee would need to be metered per-packet in order to allow for maximum flexibility — this way, a machine could transact with any miner, even just to send or receive a single packet without having previously established a relationship with that miner.

**All Transactions Occur On-Chain** Helium is built on the philosophy that all transactions should occur *on-chain*; that is, blocks should be sized and mined with a frequency such that every transaction which occurs on the network should be stored in the blockchain. To accomplish this goal the cost of mining must be low, blocks must be large enough to encapsulate a large number of transactions, and frequent enough that transactions are processed quickly.

**Allow Machines to Persist Data to the Blockchain** Because the Helium blockchain services a specific use, the `DMN`, blocks must additionally be able store fingerprints of data sent from machines along with the transaction which pays a miner for their transport service. We believe that this holistic *tamper-proof* data trail will enable entirely new use cases where the authenticity and veracity of sensor data is critical.

### 5.2 Limitations of Existing Solutions

Now that we have discussed the requirements of transactions within Helium, we outline the existing solutions for micropayments on a blockchain and address their shortcomings as they apply to our network.

**Heavyweight Transactions** This first option is suitable only for larger transactions as the service fee is smaller than the payment. This method does not work well for very small transactions as whoever pays the transaction fee ends up potentially paying more for the transaction fees than the value being exchanged. This is a similar problem to buying small-value items using credit cards today. The vendor pays a minimum fee on each credit card transaction, and under a certain charge they lose money on the transaction. These heavyweight transactions are clearly not suitable for use as a micro transaction system within Helium.

**Zero-fee Transactions** While highly desirable from a machine perspective, a true zero-fee blockchain would be fraught with spam transactions. It would be trivial to write a script to pollute the blockchain with transactions meant only to waste space on the blockchain and increase congestion on the network. Some ostensibly zero-fee blockchain implementations solve this issue in clever ways, such as offloading the work of processing and verifying transactions to the transactors themselves. However, these implementations have their own issues, for example IOTA [24] has not yet proved it is capable of operating this type of system without the need for a centralized coordinator.

**State Channels** State channels [31] allow two parties to exchange value, usually in small increments at a time, with very limited risk. If one party thinks the other is acting dishonestly, they can publish the final transaction in the

state channel to the blockchain and close the channel. At most one payment is usually at risk. However, there are several downsides: the payer has to lock up significant funds for the lifetime of the state channel, meaning they may be unable to open state channels with other parties or pay other dues; transactions in the state channel do not appear on the main chain at all; and these implementations are relatively complex to execute well (note that neither Lightning [29] nor Raiden [30] have become widely used yet).

**Payment in Arrear** Payment in arrear, after the services have been rendered, is an extremely risky method in a decentralized pseudo-anonymous system. There is no mechanism to gain certainty around the intent or honesty of the entities transacting, nor do you know if the entities control the requisite funds when the debt comes due. This model only works when the parties involved trust each other, or have some other recourse to recover funds.

### 5.3 Types of Fees in Helium

In this section we outline the types of fees needed on the Helium network, and propose solutions which take advantage of the unique characteristics of the Helium Consensus Protocol [Section 6].

#### 5.3.1 Transport Fees

Machines using the Helium network to send and receive data to and from the Internet must pay miners what is known as a *transport fee*. This fee compensates the miner for delivering data packets between the machine and the intended router on the Internet, and is unrelated to the *transaction fee* that miners earn for mining transactions as part of blocks that are recorded to the blockchain. The fee is negotiated between the router to which the machine belongs, and the miner, as machines are not directly connected to the blockchain.

Miners set the price they are willing to accept to transport data to and from the Internet on a per-byte basis.

A machines router pays miners the transport fee on transmission or reception of the data. This means that the miner will receive the transport fee prior to the transaction being mined in a block and recorded into the blockchain. This entails some risk for the miner, as they must believe that the transport payment is not malicious or fraudulent prior to it being confirmed in the blockchain. However, given how low the per-byte transport amount is likely to be, this risk seems tolerable. A miner can *blacklist* a machine or organization address if they continually abuse the system.

An example transport fee process is as follows:

1. A miner $M$ hears a packet $P$ broadcast by machine $D$
2. $M$ uses the address of $D$, attached to $P$, to identify a router $R$ as the owner of $D$
3. $M$ sends the signature $K(P)$ of $P$ and an offer of $n$ tokens for transport to $R$
4. $R$ receives $K(P)$ and the payment offer and determines if it accepts the packet for the offered price
5. Assuming $R$ accepts the packet at the offered price, it constructs a transaction $T$ of value $n$ payable to $M$ and sends it to the miner
6. Once $M$ sees the transaction in the reply it delivers $P$ to $R$ and submits $T$ to the consensus group for inclusion in the blockchain

#### 5.3.2 Transaction Fees

Transaction fees are an essential part of most blockchain implementations. They incentivize miners to include a transaction in their draft block and ensure that spam transactions do not pollute the blockchain and network.

To determine the appropriate fee for a new transaction, the transactor will take the median of the past $\delta$ packet transport fees, within some margin of error. Until $\delta$ packet transports have occured on the network, the fee will be fixed at a constant value $\alpha$. By anchoring the transaction fee to the current fees being charged for transport on the network, we root them in reality. Helium's primary purpose is to facilitate a network of wireless Internet coverage. In order to accomplish this in the long term, all of the economics of the system must align to make it practical for the primary users to transact on the network. If one set of fees were to outstrip the other, the network would quickly lose its utility for the key user segment.

To enable miners and other light clients to determine an appropriate fee, full nodes [Section 5.5] will expose a fee suggestion API. This way resource constrained entities that do not maintain a complete copy of the blockchain will not need to compute the fee from the most recent transactions. During the block submission process, miners in the consensus group [Section 6] will verify the correctness of the block and ensure that no fee has deviated beyond the acceptable threshold of $\delta$.

Due to the censorship-resilience built into the *Helium Consensus Protocol* [Section 6], there is no incentive to include larger transaction fees. Unlike Bitcoin, where miners cherry-pick the transactions with the largest fees from their mempool to include in their blocks, Helium miners cannot see the contents of the transactions without collaborating with other members of the consensus group to decrypt them. Transactions with incorrect fees (either too high or low) will be rejected prior to the block being appended to the blockchain.

### 5.3.3 Staking Fees

The *assert_location* transaction, mentioned below [Section 5.4], has a special type of fee calculation; a *dynamic* fee. Because the Helium network reaches maximum *usefulness* at a specific density of gateways, we want the fees to incentivize the network density to be as close to that ideal as possible. To that end, the transaction fee for asserting a location can be thought of as the y coordinate on a curve with the formula:

$$y = (x - D)^4 + F$$

where $D$ is the ideal gateway density and $F$ is the unit fee for a location transaction. A sample graph of this function where $D$ = 3 and $F$ = 1 follows:

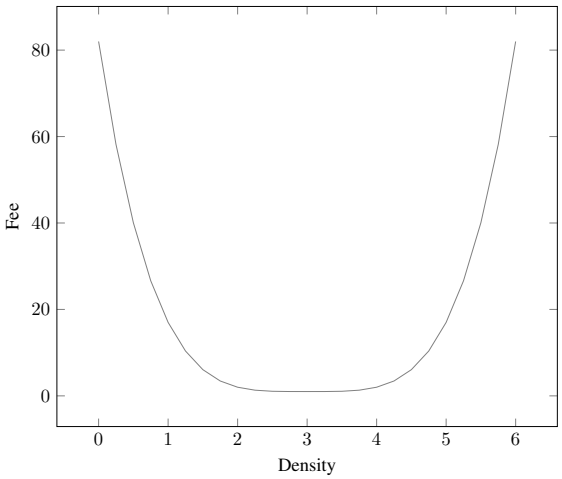


**Figure 12.** *Staking fee vs Miner density*

As can be seen, gateways near the ideal network density are cheap to add, but establishing a new network or overpopulating a network gets expensive very quickly. This serves to dis-incentivize gateway deployments that are not beneficial to the network. In particular, *Alternate Reality Attacks* and warehouses full of miners become prohibitively expensive.

Miners who have not asserted their location, and therefore not paid the staking fee, will not be considered for inclusion in the consensus group [Section 6].

Miners who move physical location will need to assert a new location, and pay the new staking fee.

## 5.4 Primitives in Helium

Having discussed the philosophy of our transaction system and presented our approach to facilitating zero-fee microtransactions on the Helium network, we now delineate the transaction primitives and their properties.

**add_gateway** Registers a new gateway on the network, adding it to an existing account which will be responsible for supplying its stake (required for mining) and will receive mining rewards [Section 6] and fees earned by the gateway

| Property | Description |
|---|---|
| gateway_address | the public key address of the gateway being added to the network |
| owner_address | the address of the owner account |
| signatures | mutual signatures of the owner and gateway |

**assert_location** Asserts a gateway's location in the form of geographic coordinates, requiring a dynamic stake

| Property | Description |
|---|---|
| gateway_address | the address asserting its location |
| nonce | a monotonically increasing integer |
| latitude | the latitude of the gateway |
| longitude | the longitude of the gateway |
| altitude | the altitude of the gateway |
| signature | the signature of the gateway |

**payment** Moves tokens from one account, the *payer*, to another account, the *payee* including the requisite fee.

| Property | Description |
|---|---|
| payer_address | the address of the sender |
| payee_address | the address of the recipient |
| nonce | a monotoically increasing integer |
| value | an integer-based representation of the tokens to send |
| signature | the signature of the sender |

## 5.5 Light Clients & Full Nodes

Until now, we have discussed how to deal with microtransactions in a cost-effective way, however we have not yet addressed how to deal with the inevitable continuously increasing size of the blockchain. One requirement for Helium is that all transactions occur on-chain. This means that the size of the full blockchain will eventually grow quite large. This is compounded by the fact that all miners on the network are gateway devices, relatively limited in computation power and storage space.

We solve this constraint by allowing mining nodes to operate as *light clients* on the blockchain, pruning old blocks and transactions as needed and keeping only the latest ledger values. They will communicate over the peer-to-peer network with *full nodes* which maintain a complete history of the blockchain to verify transactions.

This raises a question: who is responsible for operating full nodes, and what is their incentive to do so? Routers are

software-only applications with access to scalable, cloud-based storage and will be required to operate full nodes in order to fulfill their purpose. Helium Systems Inc will operate a set of hosted routers that will make it easy for developers to launch products without needing to deploy their own router. However, many enterprise developers, who are required to maintain a higher standard of privacy, will want to host their own router. Together, these routers will form a network of full nodes capable of supporting resource constrained gateways and wallets operating light clients.

## 6. Helium Consensus Protocol

Instead of an extremely computationally expensive and power hungry *Proof-of-Work*, Helium miners generate *Proofs-of-Coverage* [Section 3]. In this section we present how these useful proofs can be used to create permissionless network consensus.

### 6.1 Motivation

Many current generation blockchains rely on a computationally difficult *Proof-of-Work* to protect the network against sybil attacks, also known as *Nakamoto Consensus*. The fact that the *Proof-of-Work* is computationally expensive to create, but cheap to verify means that in order to propose a new valid block to the network there is evidence that a significant amount of computation has been expended. Due to the fact that computation is limited by hardware cost, power cost, physical space and computational efficiency of modern technology, sybil attacks become impossible. However, this approach, while fundamental to the mainstream adoption of blockchain technology, has several downsides. Chief among the downsides is the power consumption; it is estimated that the Bitcoin network is consuming more power than many small countries. Bitcoin's PoW is so wasteful it is now on the list of the top uses of electricity in the world and whenever the value of Bitcoin goes up, so do the resources devoted to mining it.

Related to the power problem is the mining pool problem. Many blockchains have mining pools where users band together to, in parallel, mine a single block and listing the pool's address as the party to get paid. The pool then shares the block reward with the members of the pool. This ends up defeating many of the advantages of decentralization as both Bitcoin and Ethereum have come to be dominated by less than 10 mining pools each. These large pools effectively prevent independent parties from mining blocks on their own. This means that the consensus protocol for these blockchains is effectively controlled by a very small number of mining pools and risks becoming further centralized.

More recently there has been increased momentum around making blockchain consensus protocols less wasteful and more useful to the network. Filecoin [25] has a *Proof-of-Spacetime* and Ethereum [5] is moving towards a *Proof-of-Stake* [26] approach.

For Helium, we desire a consensus protocol with the following attributes:

**Permissionless** Nodes should be able to freely participate in the network without permission or approval from any other entity, as long as those nodes operate in accordance with the consensus rules.

**Extremely decentralized in nature** Network consensus should be designed such that there is no incentive available for taking advantage of macro-economic factors, such as cheaper access to electricity in certain geographies, and that simply buying more hardware in the same location is either ineffective or cost prohibitive. Additionally, it should be impossible for mining pools to form and for groups to collaborate in mining blocks.

**Byzantine Fault Tolerant** The protocol should be tolerant of Byzantine failures [27] such that consensus can still be reached as long as a threshold of actors are acting honestly.

**Based on useful work** Achieving network consensus should be *useful* and *reusable* to the network. Work performed in Nakamoto Consensus-based systems is only useful for the particular block being mined and is not otherwise useful or resuable on the network. An ideal consensus system would contain work which is both useful and reusable to the network beyond simply securing the blockchain.

**High confirmed transaction rate** Our ideal consensus protocol would be able to process a very high number of transactions per second, and once a transaction is seen in a block it would be considered confirmed. Many existing blockchains require a lengthy *settlement time* while the network achieves consensus which is not ideal in a system like Helium, which may experience a very high number of transactions and where waiting for a transaction to settle is not tenable.

**Transactions are censor-resistant** Ideally miners would not be able to censor or otherwise pick and choose transactions prior to mining them. This would not only nullify any attempts to nefariously censor transactions, but would allow for otherwise unattractive transactions (such as fixed-fee transactions) to be included in the blockchain.

The remainder of this section lays out our construction of a consensus protocol with these design goals in mind that we refer to as the *Helium Consensus Protocol (HCP)*.

### 6.2 HCP

We propose a unique consensus protocol around *Proof-of-Coverage* to capture the useful work of verifying the network

as a replacement for *Proof-of-Work*, combined with a variant of the *HoneyBadgerBFT (HBFT)* [4] asynchronous byzantine fault tolerant protocol.

### 6.2.1 HBFT

HBFT is an asynchronous atomic broadcast protocol designed to achieve optimal asymptotic efficiency, initially presented by Miller et al in 2016. In HBFT, the setting assumes a network of $N$ designated nodes with distinct well-known identities ($P_0$ through $P_{\text{N-1}}$). In our HCP instantiation this network of nodes is known as the consensus group $C$. The consensus group receive transactions as input, and their goal is to reach common agreement on an ordering of these transactions and form them into blocks to be added to the blockchain.

The protocol proceeds in rounds, where after each round, a new batch of transactions is appended to the blockchain. At the beginning of each round, the group choose a subset of the transactions in their buffer, and provide them as input to an instance of a randomized agreement protocol. At the end of the agreement protocol, the final set of transactions for this round is chosen.

HBFT relies on a *threshold encryption* scheme [28] that requires transactions be encrypted using a sharded public key, such that the consensus group must work together to decrypt it. This means that no individual node is able to decrypt or censor a particular transaction without colluding with the majority of the group.

### 6.2.2 Applying *Proof-of-Coverage* to HBFT

In Helium, miners are required to submit *Proofs-of-Coverage* to the network at an epoch $\Delta_p$. These proofs are submitted as a special type of transaction, and subsequently recorded to the blockchain. As detailed in [Section 3], Helium miners increase their *score* as they submit valid proofs to the network. At an epoch $\Delta_c$ the highest scoring miners, $N$, are elected as the new HBFT consensus group $C$.

By using *Proof-of-Coverage* to elect the members of $C$ we are essentially substituting for well-known identities in the HBFT protocol. As we desire a *permissionless* network, we can use *Proofs-of-Coverage* to determine whether miners are acting honestly and reward the *most honest* miners at a given epoch by electing them to the HBFT consensus group.

### 6.2.3 The consensus group

During $\Delta_c$, the currently elected consensus group is responsible for creating blocks and appending them to the blockchain. All new transactions on the network are submitted to the current members of the consensus group. New blocks are created by $C$ at a fixed interval $\Delta_b$ and recorded to the blockchain. A token block reward is split among the members of $C$ for every block submitted, along with the sum of all fees contained within valid transactions. In the unusual case that there are no transactions during $\Delta_b$, am empty block is appended to the blockchain.

### 6.2.4 The mining process

Once the consensus group $C$ has been elected for a given $\Delta_c$ epoch, a distributed key generation phase occurs to bootstrap a threshold encryption key `TPKE`. `TPKE` is a cryptographic primitive that allows any party to encrypt a transaction to a master public key `PK`, such that $C$ must work together to decrypt it. Once $f + 1$[1] correct members of $C$ compute and reveal decryption shares, $\sigma_i$, the transaction can be recovered. Once `PK` is generated via the `TPKE.Setup` function, a block containing `PK` is immediately submitted to the blockchain. Each member $N_m$ in $C$ receives a *secret key share*, $\texttt{SK}_\text{i}$, of `PK`.

Miners on the network submit new transactions $t$ to $C$. Each member of $C$ takes a random subset of the first $B$ transactions in its queue and applies the `TPKE.Enc(PK,` $t$`)` $\rightarrow e$ function and submits them to the other member of $C$. Once the members of $C$ receive at least $N - f$ $e$ they run the $\texttt{TPKE.DecShare(SK}_\text{i}\texttt{,}\ e\texttt{)} \rightarrow \sigma_i$ function to produce their decryption share. Members broadcast their $\sigma_i$ to the other members of $C$, and once $f + 1$ members have seen $\sigma_i$ shares they can proceed to the `TPKE.Dec` function using `PK`, $e$ and the $\sigma_i$ shares and attempt to decrypt the transaction. Each member of $C$ appends decrypted transactions to their own instantiation of the next block kept in a local buffer. Double-spend and other malformed transactions are removed from these blocks at this stage.

As members of the group cannot decrypt $e$ on their own, it is not possible for a transaction to be censored by a given member prior to its inclusion in the candidate block without $f + 1$ members of $C$ colluding as transactions are received. Any honest member of $C$ that has $t$ in the first $B$ of its transaction queue will eventually be able to include $t$ in a block as the other members of $C$ can't decrypt the transaction until it has been agreed to, at which point it is too late to censor it. As the members of $C$ for a $\Delta_c$ epoch are selected based on their submitted *Proofs-of-Coverage*, making the members unpredictable, this type of collusion would be extremely challenging to execute.

Once $f + 1$ nodes have agreed on the transactions for the block, a `TPKE` threshold signature is obtained over the block. This certifies that enough nodes to exceed the Byzantine fault threshold have agreed on a block. Members of $C$ which are censoring or disagreeing on the contents of the block will produce an incompatible signature share that cannot be used

[1] $f$ is a protocol parameter equal to the number of tolerable byzantine faults

to count towards the signature threshold. This block is then gossiped out over the peer-to-peer network to all miners and added to the blockchain.

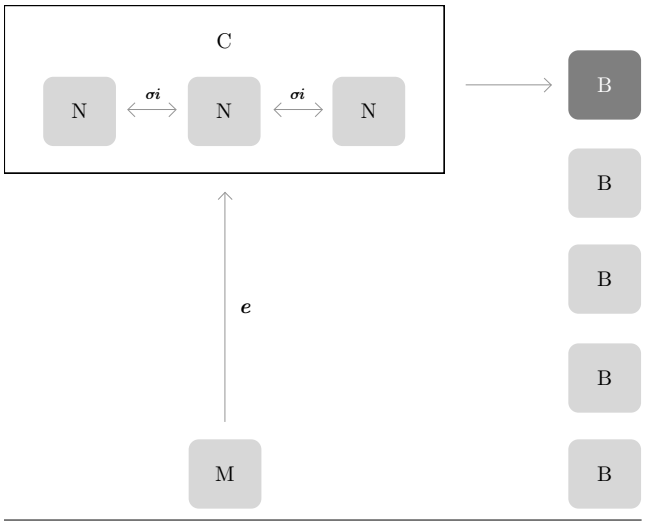


**Figure 13.** *The Consensus Group & Mining*

#### 6.2.5 Conclusion

We have presented our consensus protocol which combines a modern, asynchronous and highly efficient Byzantine Fault Tolerant consensus protocol with a novel mechanism for substituting permissioned identity with a useful and reusable *Proof-of-Coverage*. The resultant protocol satisfies the design requirements of being permissionless, decentralized, byzantine fault tolerant, based on useful work, and with a very high-rate censor proof transaction mechanism.

We refer the interested reader to [4] for a detailed breakdown and analysis of the HoneyBadgerBFT protocol.

## 7. Future Work

This paper presents a well thought-out design for building the Helium network. However, we consider this to be just the beginning of the engineering, research and design of decentralized wireless networks. We believe that this tight integration of real-world hardware with a blockchain and a native token is a novel and valuable innovation that can be applied to other kinds of networks and wireless physical layers. We believe that the future of blockchains is not about who has the most hashing power or access to the cheapest electricity, but about blockchains where the mining proof is tied to providing a valuable, verifiable service.

There are several initiatives that we either have or intend to undertake, including:

- Investigate the applicability of applying these ideas to other physical layers such as WiFi, Bluetooth and Cellular
- Explore the potential for the delivery of 5G 60GHz+ mmWave connectivity through a similar design
- Research and implement more *Proofs-of-Coverage* to keep the network secure as it grows
- Game theoretical analysis of the incentive system
- Formally prove the scoring algorithm used in the *Proof-of-Coverage*
- Create and release the *WHIP* wireless specification
- Manufacture gateways and machine modules for availability at network launch
- Investigate the deployment of a smart contract environment beyond the basic `DMN` primitives
- Continued work and evolution of Forward Error Correction techniques

## Acknowledgments

This document is the result of collaborative work by members of the Helium team, and would not have been possible without the help, feedback, and review of the board of directors, advisors, investors and collaborators of Helium. We extend our most heartfelt thanks to all involved.

We would also like to extend our thanks to Jeremy Rubin of the MIT Digital Currency Initiative. Your earliest feedback and direction was critical to some of the design decisions and evolution of this project. We also thank the Blockchain at Berkeley team for their help and detailed review of this work.

We would also like to acknowledge many of the prior works and inventions that have allowed us to create this project, most notably Bitcoin [9] and Ethereum [5].

## References


[1] Marcus Torchia, Monika Kumar. *IDC - Worldwide Semiannual Internet of Things Spending Guide*, 2017 (document)

[2] Shawn Fanning. *Napster - independent peer-to-peer file sharing*, 1999 1

[3] Mehmet Adalier. *Efficient and Secure Elliptic Curve Cryptography Implementation of Curve P-256* 2.6

[4] Andrew Miller and Yu Xia and Kyle Croman and Elaine Shi and Dawn Song. *The Honey Badger of BFT Protocols*, 2016 2.2, 6.2, 6.2.5

[5] Vitalik Buterin. *Ethereum*, 2014 3.1, 6.1, 7

[6] LoRa Alliance. *LoRa Alliance - Wide Area Networks for IoT*, 2013 2.4.1

[7] Ingenu. *RPMA Technology* 2.4.1

[8] IEEE. *IEEE Standard for Low-Rate Wireless Networks*, 2015 2.4.1

[9] Satoshi Nakamoto. *Bitcoin: A peer-to-peer electronic cash system*, 2008 3.1, 7

[10] E. W. Dijkstra. *A note on two problems in connection with graphs*, 1959 4, 3.3.4

[11] David Karger, Eric Lehman, Tom Leighton, Matthew Levine, Daniel Lewin, Rina Panigrahy. *Consistent Hashing and Random Trees: Distributed Caching Protocols for Relieving Hot Spots on the World Wide Web*, 1997

[12] Adam Langley, Google. *Roughtime - a project that aims to provide secure time synchronisation* 3.4

[13] Mehmud Abliz, Taieb Znati. *A Guided Tour Puzzle for Denial of Service Prevention*, 2009 3.2

[14] Mobile Experts *Asset Tracking IoT Devices*, 2017 4.1

[15] Guofang Dong, Bin Yang. *TDOA-Based and RSSI-Based Underground Wireless Positioning Methods and Performance Analysis* 4.2.1

[16] Mohamed Laaraiedh, Lei Yu, Stephane Avrillon. *Comparison of Hybrid Localization Schemes using RSSI, TOA, and TDOA*, 2011 4.2.1

[17] Mohamad Yassin, Elias Rachid, Rony Nasrallah. *Performance Comparison of Positioning Techniques in Wi-Fi Networks*, 2014 4.2.1

[18] Muhammad Farooq-i-Azam, Muhammad Naeem Ayyaz. *Location and Position Estimation in Wireless Sensor Networks*, 2016 4.2.1

[19] Sangdeok Kim, Jong-Wha Chong. *An Efficient TDOA-Based Localization Algorithm without Synchronization between Base Stations*, 2015 4.2.2

[20] Igor Olegovych Tovkach, Serhii Yakovych Zhuk. *Recurrent Algorithm for TDOA Localization in Sensor Networks*, 2017 4.2.2

[21] Peter W. Boettcher, Gary A. Shaw. *A Distributed Time-Difference of Arrival Algorithm for Acoustic Bearing Estimation* 4.2.2

[22] Shuai He, Xiaodai Dong, Wu-Sheng Lu. *Asynchronous Time Difference of Arrival Positioning System*, 2015 4.2.2

[23] LoRa Alliance. *LoRaWAN - LoRa Alliance Technology*, 2014 4.2.1

[24] David Snsteb, Sergey Ivancheglo, Dominik Schiener, and Serguei Popov. *IOTA - Next Generation Blockchain*, 2015 5.2

[25] Protocol Labs. *Filecoin - A decentralized storage network*, 2017 6.1

[26] Wikipedia. *Proof-of-Stake* 6.1

[27] K Driscoll, B Hall, M Paulitsch, P Zumsteg, H Sivencrona. *The Real Byzantine Generals*, 2004 6.1

[28] Joonsang Baek, Yuliang Zheng. *Simple and Efficient Threshold Cryptosystem from the Gap Diffie-Hellman Group*, 2003 6.2.1

[29] Joseph Poon, Thaddeus Dryja. *Lightning Network - Scalable, Instant Bitcoin/Blockchain Transactions*, 2017 5.2

[30] brainbot. *The Raiden Network - Fast, cheap, scalable token transfers for Ethereum*, 2017 5.2

[31] Jeff Coleman. *State Channels*, 2015 5.2