Helium

A Decentralized Wireless Network

Amir Haleem Andrew Allen Andrew Thompson Marc Nijdam Rahul Garg

Helium Systems, Inc.

Release 0.4.2 (2018-11-14)

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.

The Helium network is a *decentralized wireless network* that enables devices 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 Helium 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, we inject 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.

Our secure and open-source primitives enable developers to build low-power, Internet-connected devices quickly and cost-effectively. The Helium network has a wide variety of applications across industries and is the first decentralized wireless 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 one of our founders 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.

The Helium network is a wide-area wireless networking system, a blockchain, and a protocol token. The blockchain runs on a new consensus protocol, called the Helium Consensus Protocol, and a new kind of proof, called *Proof-of-Coverage*. The Miners who are providing wireless network coverage in a cryptographically verified physical location and time submit proofs to the Helium 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 Devices and the Internet via a network of independent providers that does not rely on a single coordinator, where: (1) Devices 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 Helium 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 our GitHub repo at `https://github.com/helium` as we continue to open-source various components of the system over time.

1.1 Key Components

The Helium network 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.

Helium Network We demonstrate an entirely new purpose-built blockchain network built to service WHIP and provide a system for authenticating and identifying devices, providing cryptographic guarantees of data transmission and authenticity, offer transaction primitives designed around WHIP, 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*.

WHIP We introduce a new open-source and standards-compliant wireless network protocol, called *WHIP*, designed for low power Devices 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 Device using WHIP without the need for expensive and power-hungry satellite location hardware. Devices can make immutable, secure, and verifiable claims about their location at a given moment in time which is recorded in the blockchain.

DWN We present a decentralized wireless network (DWN) that provides wireless access to the Internet for Devices by way of multiple independent Miners and outlines the Helium network and WHIP specification by which participants in the Helium 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 Device data to the Internet.

1.2 System Overview

- The Helium network is a *decentralized wireless network* built around WHIP on a purpose-built blockchain with a native token.
- Devices 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.
- Devices 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 Device's data. Miners are paid once they prove they have delivered data to the Device's 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 protocol tokens 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 DWN

We introduce the core conmponents of the DWN.

2.1 Participants

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

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

Miners provide wireless network coverage to the Helium network via purpose-built hardware, called Hotspots [Section 2.5], which provide a long-range bridge between WHIP devices and the Internet. Users join the Helium network as Miners by purchasing or building a Hotspot that conforms to WHIP, 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 Device can use. Miners join the Helium 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

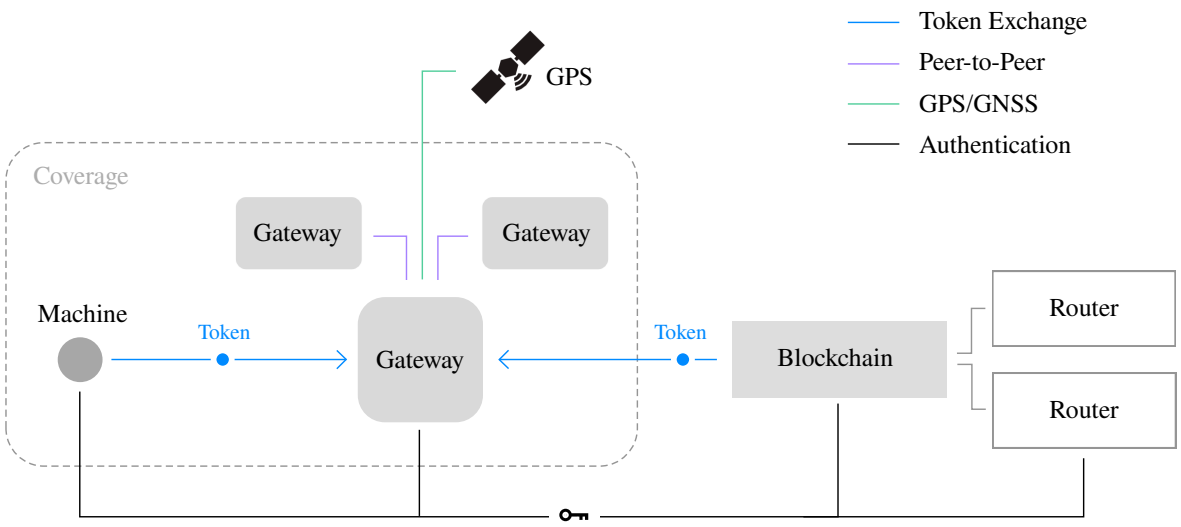

Figure 1. *System Overview*

reward and transaction fees for any transactions included in the block once mined. As a Miner's score drops their probability of being elected to the consensus group and mining blocks diminishes.

Routers are Internet applications that purchase encrypted Device 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 Device without needing satellite location hardware, which we call *Proof-of-Location*. Routers are the termination point for Device data encryption. Devices record to the blockchain to which Routers a given Miner should send their data, such that any Hotspot on the Helium network can send any Device data to the appropriate Router. Routers are responsible for confirming to Hotspots that Device data was delivered to the correct destination and that the Miner should be paid for their service.

2.2 Blockchain

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

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

The Helium network is also a *physical* wireless network instantiation. The participants in the Helium network can be thought of as follows:

WHIP The Helium 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 device and corresponding Internet-hosted router.

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

Devices exist in the form of hardware products that contain a WHIP-compatible radio transceiver and communicate with Hotspots on the Helium network. WHIP is designed to facilitate low power data transmission and reception, so typically Devices exist in the form of battery-powered sensors that can operate for several years using standard batteries (although mains-powered Devices also work quite well). Devices 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. Device 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 Devices. 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 wireless network should use non-proprietary protocols and modulation schemes and that participants in the Helium network 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 our 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 *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, devices using WHIP do not need to coordinate with Hotspots on channel selection as Hotspots 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 necessary.

Co-Existence Performance As the number of Devices 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 Hotspots.

2.4.3 Implementation

WHIP supports several data rates, channel bandwidths, and error-correction techniques. Hotspots and Devices 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 Device Network Alliance.

2.5 Hotspots

Hotspots are physical network devices operated by Miners that create wireless RF coverage over wide areas. They transmit data back and forth between Routers on the Internet and Devices on the network, process blockchain transactions, and create *Proofs-of-Coverage* for the Helium network [Section 3]. Hotspots can connect to the Internet using any TCP/IP capable backhaul, such as Ethernet, WiFi or Cellular. Each Hotspot contains a radio frontend chip capable of listening to several MHz of radio spectrum at a time and can hear all wireless 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 Hotspots can hear any Device traffic transmitted within the frequency range, and no synchronization between the Hotspot and Device needs to occur. This allows Devices to remain inexpensive and relatively simple and reduces wireless protocol overhead. If a Miner wishes to minimize their Hotspot hardware costs, synchronized frequency hopping schemes are also permitted within the specification as a cheaper alternative to a more expensive radio frontend.

Hotspots 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 Hotspot 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 Devices if multiple Hotspots observe the same packet. This allows devices 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].

We will make both a complete open-source reference design and a finished product available at launch of the Heilum network.

2.6 Devices

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

WHIP is designed such that Devices 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 Device 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 MacDevicehine, 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 Devices via Hotspots and route them to appropriate destinations such as an HTTP or MQTT endpoint.

Routers serve several functions on the Helium network, including:

- Authenticating Devices with the Helium network;
- Receiving packets from Hotspots and routing them to the Internet;
- Delivering downlink messages, including OTA updates, to Devices via Hotspots;

- 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*; and
- Storing and making available a full copy of the blockchain ledger by acting as a *full node* [Section 5.5]

When a Hotspot receives a data packet from a Device on the Helium network, it queries the blockchain to determine which Router to use given the Device's Helium network address. Anyone is free to host their own Router and define their Devices' traffic to be delivered there by any Hotspot on the Helium network. This ability allows users of the Helium 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 Device'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. We will make available an open source reference implementation of a Channel that can be used to build additional interfaces to Internet services.

We 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 Device Network Alliance.

3. *Proof-of-Coverage* and *Proof-of-Serialization*

In the Helium network, Miners must prove that they are providing wireless network coverage that Devices are able to use to communicate with the Internet. Miners do this by complying with the *Proof-of-Coverage* protocol which the Helium 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.

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 to the blockchain, the cumulative difficulty of the chain becoming prohibitively difficult to fabricate.

These computation-heavy proofs are, however, not otherwise *useful* to blockhain networks. We define useful as work that is valuable to a blockchain network beyond securing the ledger. 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 the Helium network must be resistant to *Sybil attacks* in which dishonest Miners create pseudonymous identities and use them to subvert the Helium 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 the Helium network. We must also be resistant to a new attack vector: *alternate reality attacks*, which 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 the Helium Consensus Protocol [Section 6] that uses *Proof-of-Coverage* to both secure the blockchain and provide an extremely useful service to the Helium network, which provides wireless network coverage that Devices can use to send data to and from the Internet.

3.2 Inspiration

Proof-of-Coverage is an innovative proof that allows Miners to prove that they are providing wireless network coverage W in a specific region to a challenger, C. Proof-of-Coverage 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. Proof-of-Coverage 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 Proof-of-Coverage we aim to solve for the following:

- Prove that Miners are operating RF hardware and firmware compatible with WHIP;
- Prove that Miners are located in the geography they claim by having them communicate via RF; and
- 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 the Helium network as RF networking has limited range and therefore cannot communicate with peers anywhere on the Helium 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 Helium 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, then clients end up with cryptographic proof of that behavior.

3.3 Constructing *Proof-of-Coverage*

With 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; and
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 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.

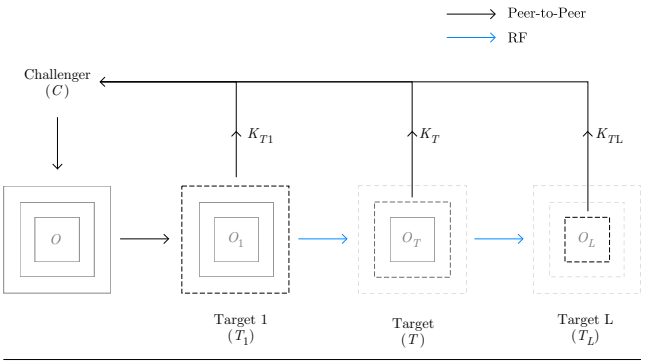

Figure 2. *Multi-Layer Data Packet Deconstruction*

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 Helium network. T does not need to be geographically proximate to C. To locate T, C initially seeds verifiable entropy, η, into the selection process by signing the current block hash with its 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 η. 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 C 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 Helium network and received by geographically proximate targets T_n.

Geographically proximate is defined as within a radius of T, a network value T_{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 Elliptic-Curve Diffie-Hellman (ECDH) derived symmetric key, S is a nonce, ψ 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_{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 ECDH exchange to compute a shared secret, known only to both parties C and T_L; and
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].

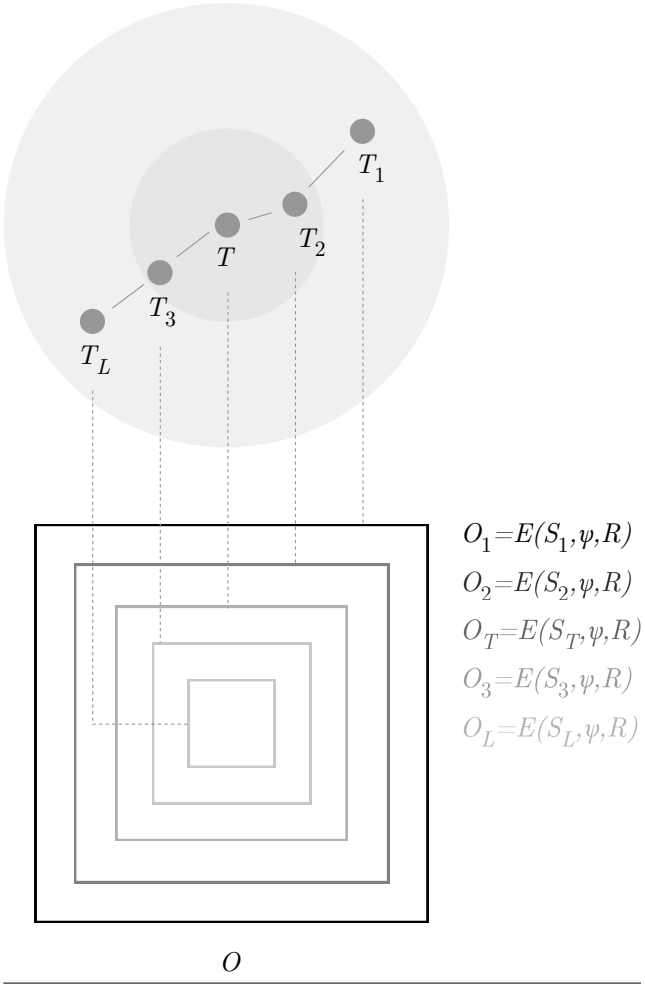

Figure 3. Construction of O

3.3.3 Creating the Proof

Once O has been constructed, it is delivered to T_1 via the Helium network and immediately broadcast by T_1 via the Helium network. WHIP is not a point-to-point system, so several Miners within proximity of T_1 will hear O. In this example, only the specific target T will be able to decrypt E and send a valid receipt back to the challenger, C.

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

1. T_1 receives O from C via the Helium network, decrypts the outermost layer and immediately broadcasts it R via the Helium network;
2. T hears O and attempts to decrypt the value of E by using its private key where $pk: \ E_{pk}\left(S, \psi, R\right)$;
3. T records both the time of arrival β and the signal strength υ of O;
4. If successful, T then creates signed receipt K_s, where $K_s = (S||\beta||\upsilon)$ signed by the private key of T;
5. T submits K_s to C via the Helium network, removes the outer most layer, and wirelessly broadcasts the remainder O; and
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 λ, 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 β and υ 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 β is limited by the speed of light τ 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 τ multiplied by the geographical dis-

tance D plus some small episilon value, $\upsilon = \tau \times \Big(D + \epsilon\Big)$. For υ, because of the inverse-square law, we can calculate the maximum RSSI (Received Signal Strength Indication) possible for a packet transmitted, μ, from $T_g - 1$ to T_g as $\mu = \frac{1}{D^2}$. Hotspots that are closer than expected, or which are transmitting at a higher power to mask their location disparity, are unlikely to get μ 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 λ 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 Helium network.

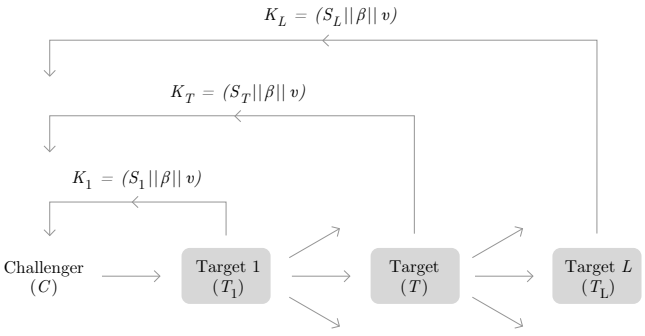

Figure 4. *Proof-of-Coverage* flow

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 Helium Consensus Protocol described in [Section 6]. When Miners join the Helium network, they are assigned a score, ϕ_m. We consider any Miner with a score greater than ϕ_m to be an *honest miner*. This score depreciates according to the number of verifications the Miner has as well as the height since its last successful verification. As ϕ_m decreases the probability of the Miner M being the target for C increases, such that the Helium network continually attempts to prove that the lowest scoring Miners are acting honestly, and giving Miners a reasonable chance to improve their scores.

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

M,	Miner
υ,	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:

υ',	$\upsilon/10.0$
h',	$h/480$

Using the above we can now construct a staleness-factor, δ, 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 υ indicates that the Miner is consistently failing verification.
2. If υ = 0, then we do not have any trust information, therefore, we use a steep parabolic curve for the decay dependent on h'.
3. If $\upsilon > 0$, then 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 υ 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 $\upsilon < 0$, then 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 υ, 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.

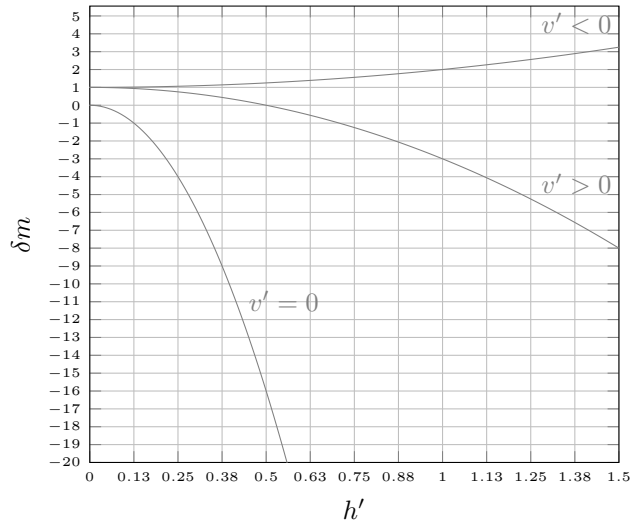

Figure 5. *Trendlines for the scoring functions*

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:

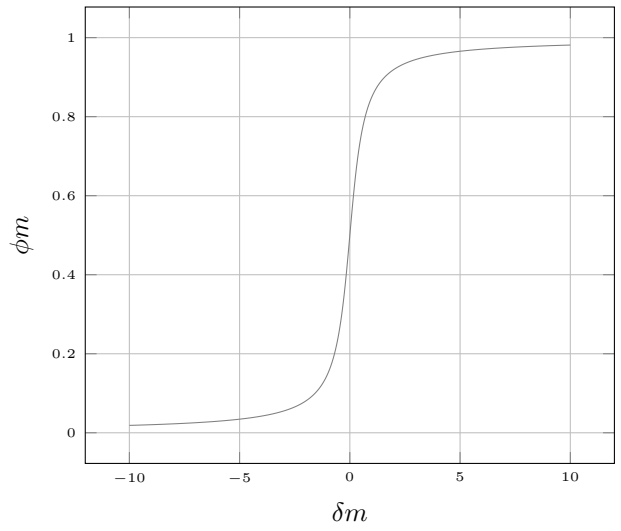

Figure 6. *Scoring algorithm and the resulting staleness factor*

[Figure 7] shows a snapshot of a random subset of the Helium 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].

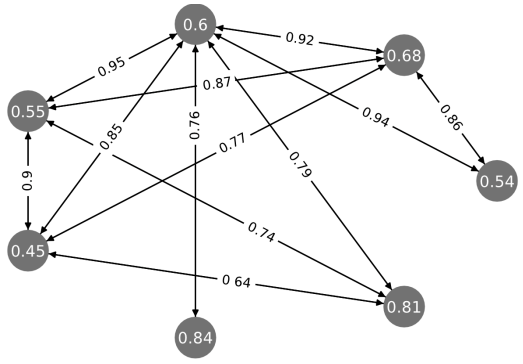

Figure 7. *Snapshot of a random subset of the initial network*

After 10,000 iterations the Helium 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.

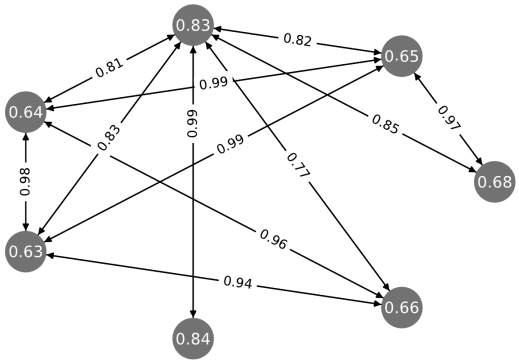

Figure 8. *Snapshot of a random subset of the network after 10000 iterations*

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 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 λ 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 Helium 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 η 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 η 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; and
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, with whom to prove contact serialization;
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*, where $K = H\left(R||M_1||M_2\right)$;
5. M sends K to M_1. M_1 replies with T, a signed message including the current time T_1 and K; and
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; and
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 of the correct time 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 the correct time, T_t.

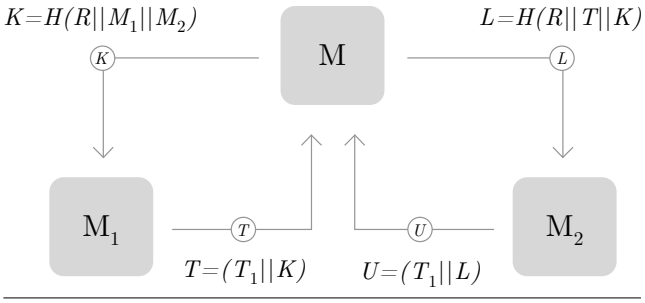

Figure 9. *Creating Proof-of-Serialization*

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 M_1, 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 $K = H(R||M_1||M_2)$;
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, $L = H(R||T||K)$, 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; and
7. T_n can then be confirmed to be T_t, the correct time

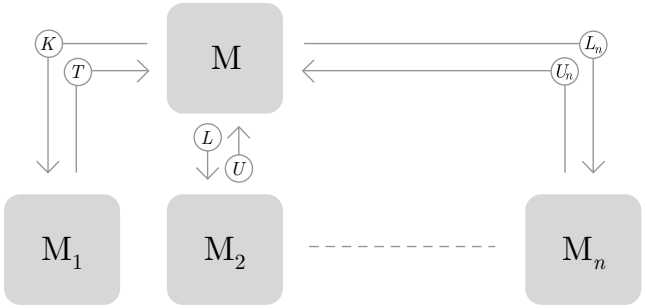

Figure 10. *Verifying Proof-of-Serialization*

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, η, 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 Miners. We can take advantage of these proofs to determine the physical *geolocation* of WHIP-compatible Devices and generate a new type of proof based on the Devices geolocations. We call this *Proof-of-Location*.

4.1 Motivation

Location tracking is one of the most valuable use cases for low power Devices. 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 Devices 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 extraordinarily 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 Devices that the Helium network 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 Device 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.

4.2 Constructing *Proof-of-Location*

Our goal is to verify the physical geolocation of a given Device, 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 received 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 Devices emit, and synchronize the clocks of Miners on the Helium network.

An example timestamping flow is as follows:

1. A Device, D, broadcasts a packet P containing arbitrary data via the Helium 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 Hotspot radio frontend;
4. A signed transaction including P and T_n are delivered to the router R belonging to D by M_n; and
5. R has now received several copies of P, each of which has a slightly varying value of T_n

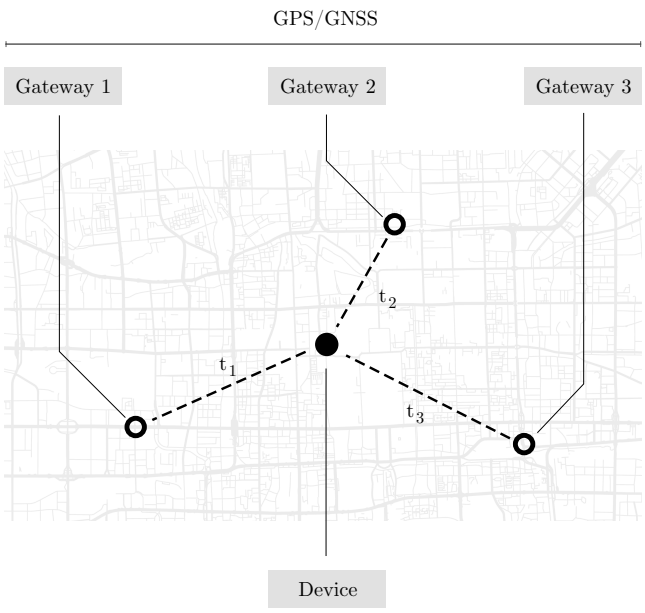

Figure 11. *Geolocation via TDoA*

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 Miner's radio hardware and a fast enough processor to sample this data, identify an appropriate packet, and record the timestamp. Typically, a *Field Programmable Gate Array (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, our Hotspot 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 our Hotspot will be released as open source software at launch of the Helium network.

4.2.2 Using timestamps to derive location

Now that the Devices Router, R, is in possession of a variety of signed messages, which include the precise timestamps, T_n, it is possible to *solve* for the location of the Device 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 whitepaper.

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 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; and
- 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 Device 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 network provide functionality that enables address-to-address transfers of protocol 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 `DWN`. We will first address Helium's need for microtransactions and propose a new solution.

5.1 The Helium Nework's Need for Microtransactions

Devices Pay Per Packet The goal of the Helium network is to offer Internet data transport fees (the fees paid by Devices 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 Device 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 The Helium network 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 Helium 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 blocks must be created frequently enough that transactions are processed quickly.

Allow Devices to Persist Data to the Blockchain Because the Helium network services a specific use, the `DWN`, blocks must additionally be able store fingerprints of data sent from Devices along with the transaction, which pays a Miner for its 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 the Helium network, we outline the existing solutions for micropayments on a blockchain and address their shortcomings as they apply to the Helium 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 the Helium network.

Zero-fee Transactions While highly desirable from a device 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, it 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 that take advantage of the unique characteristics of the Helium Consensus Protocol [Section 6].

5.3.1 Transport Fees

Devices 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 Device 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 Device belongs, and the Miner, as Devices 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 Devices 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 Device 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 Device 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; and
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 Helium network

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 Helium network.

To determine the appropriate fee for a new transaction, the transactor will take the median of the past δ packet transport fees, within some margin of error. Until δ packet transports have occured on the Helium network, the fee will be fixed at a constant value α. By anchoring the transaction fee to the current fees being charged for transport on the Helium network, we root them in reality. The Helium network'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 Helium network. If one set of fees were to outstrip the others, then the Helium 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 δ.

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 Hotspots, we want the fees to incentivize the Helium 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 Hotspot density and F is the unit fee for a location transaction. A sample graph of this function where D = 3 and F = 1 follows:

As can be seen, Hotspots 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 Hotspot 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.

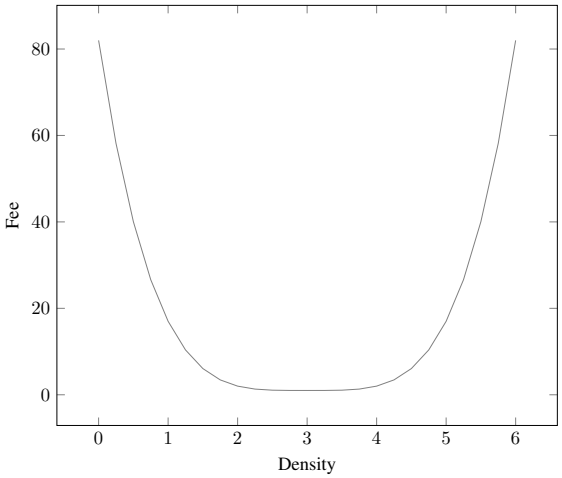

Figure 12. *Staking fee vs Miner density*

5.4 Primitives in The Helium Network

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

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

Property	Description
hotspot_address	the public key address of the Hotspot being added to the network
owner_address	the address of the owner account
signatures	mutual signatures of the owner and Hotspot

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

Property	Description
hotspot_address	the address asserting its location
nonce	a monotonically increasing integer
latitude	the latitude of the Hotspot
longitude	the longitude of the Hotspot
altitude	the altitude of the Hotspot
signature	the signature of the Hotspot

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 and 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 the Helium network 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 Helium network are Hotspot 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. We 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 Hotspots and wallets operating light clients.

6. Helium Consensus Protocol

Instead of an extremely computationally expensive and power hungry *Proof-of-Work*, 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 Helium 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 Helium 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 Proof-of-Work 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 the Helium network, we desire a consensus protocol with the following attributes:

Permissionless Nodes should be able to freely participate in the Helium 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 reusable on the network. An ideal consensus system would contain work that 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 the Helium network, 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*.

6.2 Helium Consensus Protocol

We propose a unique consensus protocol around *Proof-of-Coverage* to capture the useful work of verifying the Helium 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 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 receives transactions as input, and its 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 chooses a subset of the transactions in its buffer and provides 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 the Helium network, miners are required to submit *Proofs-of-Coverage* to the Helium network at an epoch, Δ_p. These proofs are submitted as a special type of transaction, and subsequently recorded to the blockchain. As detailed in [Section 3], Miners increase their *scores* as they submit valid proofs to the Helium network. At an epoch, Δ_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 Δ_c, the currently elected consensus group is responsible for creating blocks and appending them to the blockchain. All new transactions on the Helium network are submitted to the current members of the consensus group. New blocks are created by C at a fixed interval Δ_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 Δ_b, an empty block is appended to the blockchain.

6.2.4 The mining process

Once the consensus group C has been elected for a given Δ_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, σ_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 Helium 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 $\texttt{TPKE.Enc(PK, } t) \rightarrow e$ function and submits them to the other member of C. Once the members of C receive at least $N - f$ e they run the $\texttt{TPKE.DecShare(SK}_\text{i}, e) \rightarrow \sigma_i$ function to produce their decryption share. Members broadcast their σ_i to the other members of C, and once $f + 1$ members have seen σ_i shares they can proceed to the `TPKE.Dec` function using `PK`, e and the σ_i shares and attempt to decrypt the transaction. Each member of C appends decrypted transactions to its 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, a given member cannot censor a transaction 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 cannot 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 Δ_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 that are censoring or disagreeing on the contents of the block will produce an incompatible signature share that cannot be used to count towards the signature threshold. This block is then gossiped via the Helium network to all Miners and added to the blockchain.

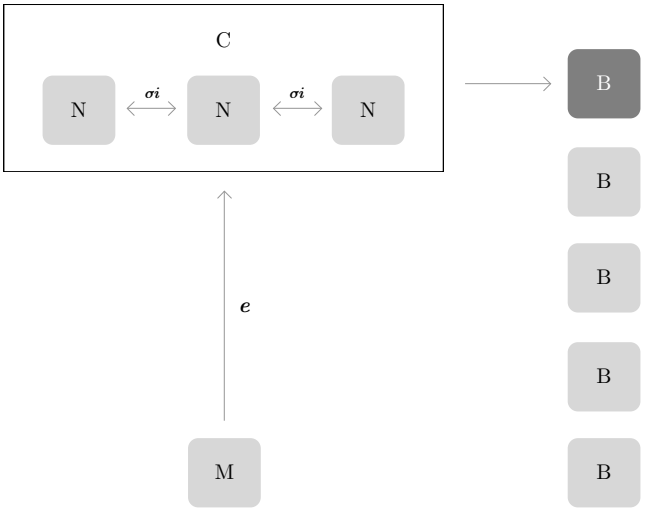

Figure 13. *The Consensus Group and Mining*

6.2.5 Conclusion

We have presented the Helium 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

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

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 Helium 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 Hotspots and Device modules for availability at launch of the Helium network
- Investigate the deployment of a smart contract environment beyond the basic `DWN` primitives
- Continued work and evolution of Forward Error Correction techniques

Acknowledgments

This document is the result of collaborative work by members of our team, and would not have been possible without the help, feedback, and review of our board of directors, advisors, investors and collaborators. 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