# Marconi Protocol

Marconi Foundation

May 2, 2018

Version 0.71

**Abstract**

Blockchain technology has often been proposed as a solution to the problems inherent with centralized systems. Blockchain projects are being developed to provide decentralized computing, storage, and a suite of applications to realize a decentralized future. However, these projects all continue to build atop the same underlying network infrastructure consisting of switches and routers connected by Ethernet, a foundation which remains fragile due to being insecure, difficult to manage, and centrally controlled. In this paper we describe a new protocol designed to address these challenges. Marconi is a networking and blockchain protocol that allows smart contracts for network packets. The protocol has been designed down to layer 2 of the OSI model and works with wired and wireless standards. Contributions covered in this paper include systems and methods for processing network packets using smart contracts, forming secure mesh networks with decentralized traffic auditing and metering, programmatically jump starting branch chains connected to a global blockchain, virtualizing and binding OSI layer 2 connections, ranking peer nodes, and performing decentralized distributed network routing.

## Contents

## 1 Introduction

Blockchain technology has often been proposed as a solution to the problems inherent with centralized systems. Globally, the $200 billion blockchain ecosystem consists of over 1900 projects [1] that are developing decentralized computing, storage, and a suite of applications to realize a decentralized future. But these projects all continue to build atop the same underlying network infrastructure consisting of switches and routers connected by Ethernet, a foundation which remains fragile due to three fundamental problems.

The first is that today's network infrastructure is insecure. This is because it's powered by Ethernet, a networking technology that has improved in terms of bandwidth but otherwise has gone relatively unchanged for the past 30 years. Ethernet was devised during an age unconcerned with privacy and security when the main focus was instead on connectivity [2, 3], thus Ethernet has no encryption built into its design. This exposes raw network packets, allowing internet service providers and governments to easily monitor and surveil user activity. Unfortunately, common network security protocols operate at several layers higher up the stack (see Figure 1 and note TLS and SSL occurring at OSI layer 4 and above), while Ethernet remains insecure at layer 2. Therefore these security protocols don't protect the entire network packet, leaving network traffic vulnerable to attacks like traffic pattern analysis and packet injection.

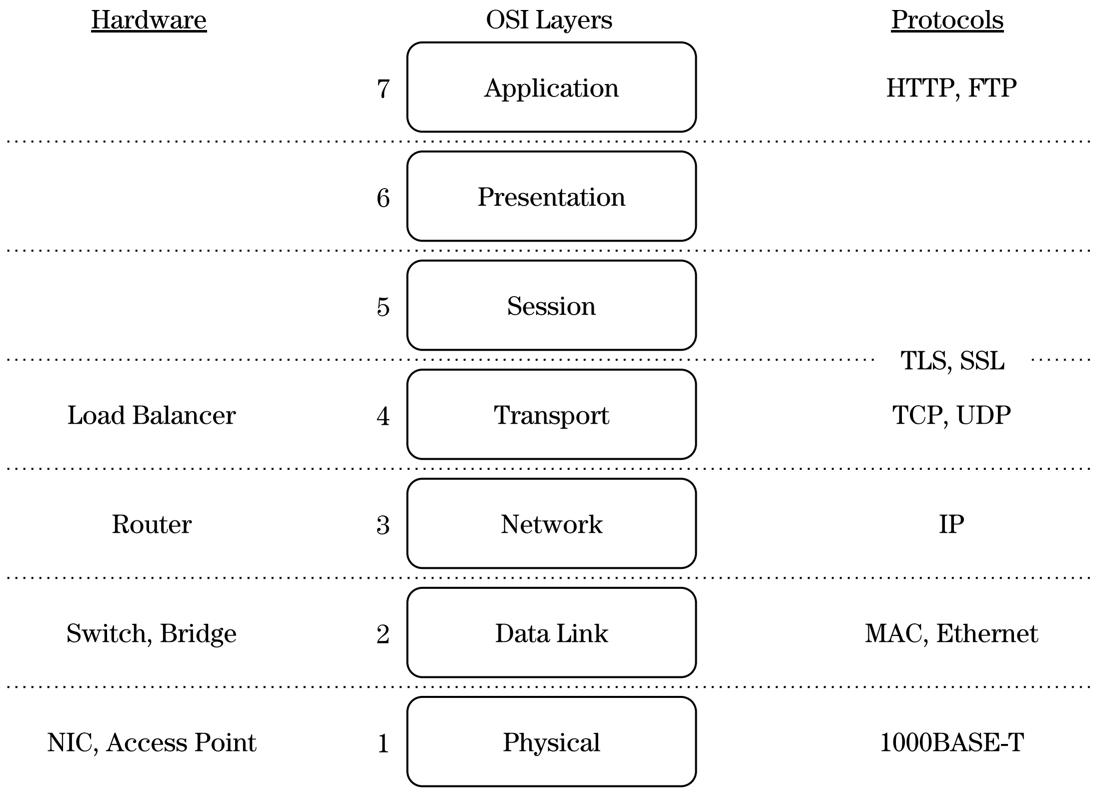

Figure 1: Hardware and protocols with relation to the OSI model

The second problem is that core network infrastructure is inflexible and difficult to manage. The switches, routers, and bridges that form the network are hardware driven and expensive to buy, configure, and maintain. Adding network capacity or new functionality like intrusion detection and prevention systems or load balancing typically requires installing new network appliances. Even upgrading existing equipment can cost considerable overhead since updates may require firmware changes which in some cases must be done on-site.

The final problem is that current network infrastructure is centrally controlled. In any given region, a small number of entities—often just one or two internet service providers—serve as the gateway for all internet traffic. When the internet service provider suffers fiber cuts, equipment failure, or purposely interrupts service to perform maintenance, all users lose internet access. Businesses get hit especially hard as loss of internet can halt operations or productivity. In addition, this monopolistic control is problematic for countries that lack net neutrality now that blockchain networks have grown to considerable sizes. As of February 1, 2018, Ethereum's blockchain data directory size is 669 gigabytes and has been growing by 416% annually [4]. As blockchain adoption continues, blockchain traffic will likely fall under the crosshairs of internet service providers. Unfortunately, this centrally controlled infrastructure that we've described is the platform on which the blockchain ecosystem depends (see Figure 2), an ecosystem where new projects are constantly emerging with the goal of decentralizing the services we use every day.

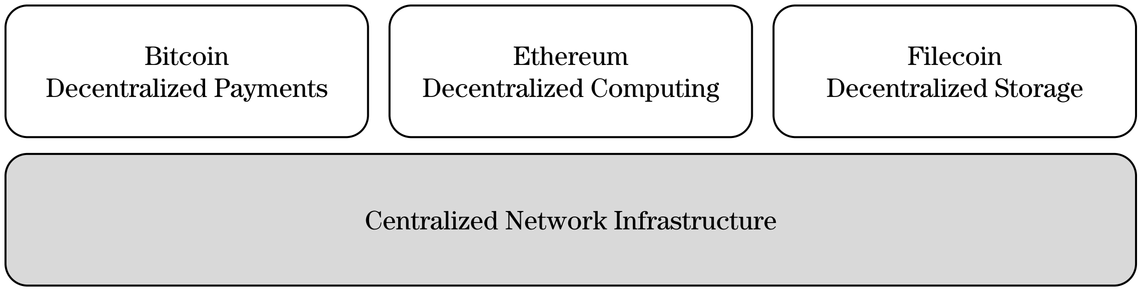

Figure 2: Dependencies on existing centralized network infrastructure

To address these challenges, we propose Marconi, a protocol that enhances and can even replace existing network infrastructure, allowing any network to realize the full benefits of decentralization. Marconi is a networking and blockchain protocol designed down to OSI layer 2 that allows smart contracts for network packets. Marconi addresses the existing challenges by:

- Securing Ethernet with packet-level encryption.
- Augmenting existing network infrastructure with smart packet contracts and programmable branch blockchains to enable dynamic network adjustment and novel security and networking applications.
- Decentralizing ownership of network infrastructure by incentivizing the formation of mesh networks where user devices provide switching, routing, and packet processing functionality.

To illustrate the capabilities of the Marconi Protocol, we provide here two example applications that can be built on top of it. The protocol's components and architecture are described in greater detail later in this paper along with additional use cases.

The first example application is for blockchain projects looking to quickly spin up their own blockchain network. This might be a brand new project or a token project looking to migrate to its own chain. More concretely, take for instance a decentralized cryptocurrency exchange. Similar to other blockchain platforms, this exchange can be built on top of the Marconi Protocol using smart contracts, but with the added benefit of being on its own dedicated blockchain powered by its own token. This is achieved by invoking a Marconi branch contract to create a programmable blockchain, which provides greater token utility and decouples performance from other projects for higher transaction throughput. Another powerful benefit is that the decentralized exchange can make use of smart packet contracts to improve network resilience and security. For example, many users of the world's largest cryptocurrency exchange recently had their account credentials stolen by a phishing attack involving URLs with unicode characters [5, 6]. This attack could have been prevented by using smart packet contracts to analyze network packets to catch these suspicious URLs. Smart packet contracts can also facilitate decentralized network administration such as re-routing packets among network nodes for better load balancing. Lastly, by taking advantage of the Marconi Protocol's decentralized network infrastructure, a decentralized cryptocurrency exchange can better protect itself from being forcibly shut down.

The second example application is for secure field networks. Using the Marconi Protocol and the technology it's built on, these networks can be rapidly deployed in battlefield or disaster relief situations with their traffic history stored on a ledger that can be audited after the network is decommissioned. While deployed, the network can be dynamically extended with wired or wireless hardware that uses the Marconi Protocol for MAC layer security on each data link. The network can also be segmented with different permissions as necessary, ensuring only certain traffic types can be sent in specific segments. With smart packet contracts, next generation firewalls and intrusion prevention systems can be implemented to protect against malicious traffic. And due to its decentralized nature, the network is fault tolerant, automatically reconfiguring to work around links or nodes that go down unexpectedly. Finally, post action analysis can be done with machine learning techniques on historical data, including sensor data from IoT devices used by personnel, to understand whether the network was compromised and to optimize for better future deployments.

# 2 Overview

The Marconi Protocol defines the rules and provides the primitives by which peers can securely connect and communicate in order to form and participate in the Marconi Network, a global construction which acts as a network of networks. Here we give an overview of these concepts and their most novel aspects, leaving the details for Section 3. As we shall see in Section 4, these different pieces ultimately fit together to enable powerful applications related to security, networking, and decentralization, such as the ability to catch phishing attempts, to obfuscate network traffic, to easily launch new blockchains, and to decentralize any network.

## 2.1 Marconi Protocol

As mentioned previously, the Marconi Protocol facilitates secure network communication, flexible network infrastructure, and the formation of mesh networks. The supporting technology can be broken down into three major components which we'll summarize here but later discuss in greater depth.

- **Marconi Pipe.** Marconi Pipes provide a secure communication channel for transporting network traffic between peers. The pipes are established all the way down to layer 2 of the OSI model and provide encryption, routing, and processing capabilities. Marconi Pipe works with wired standards to allow the Marconi Protocol to be used as an overlay on existing internet infrastructure. We also have an extension called Marconi Link which has been designed to work with wireless standards such as Bluetooth, Wi-Fi, and the U-NII radio band [7, 8] to power scalable mesh networks, both public and private.
- **Smart Packet Contracts.** Network packets can be routed and processed using smart contracts. This technology unlocks numerous use cases for smart decentralized networking applications such as anti-phishing and anti-malware protection, intrusion detection and prevention systems, and distributed virtual private networks.
- **Branch Chains.** New blockchains can be programmatically jump started by branching from a global chain, and each of these branch chains can have its own custom rules which are specified via a special type of smart contract called a branch contract. For example, if a blockchain project wants to create a new token or migrate its existing token to a dedicated chain, this can be achieved by invoking a branch chain. We also use branch chains to organize nodes participating in mesh networks, decoupling them from having a strong dependency on a global blockchain. This type of branch chain is called a mesh chain.

These components are the building blocks that underpin the Marconi Network.

## 2.2 Marconi Network

The Marconi Network enables the formation of autonomous networks between peers and globally organizes them into a network of networks (see Figure 3). Peers may be infrastructure service nodes, internet-enabled computing devices, or network end users. Marconi Network Contracts are agreements between these peers defining how much data will be exchanged, for how long, what types of smart packet contracts will be enabled, and at what fuel price.

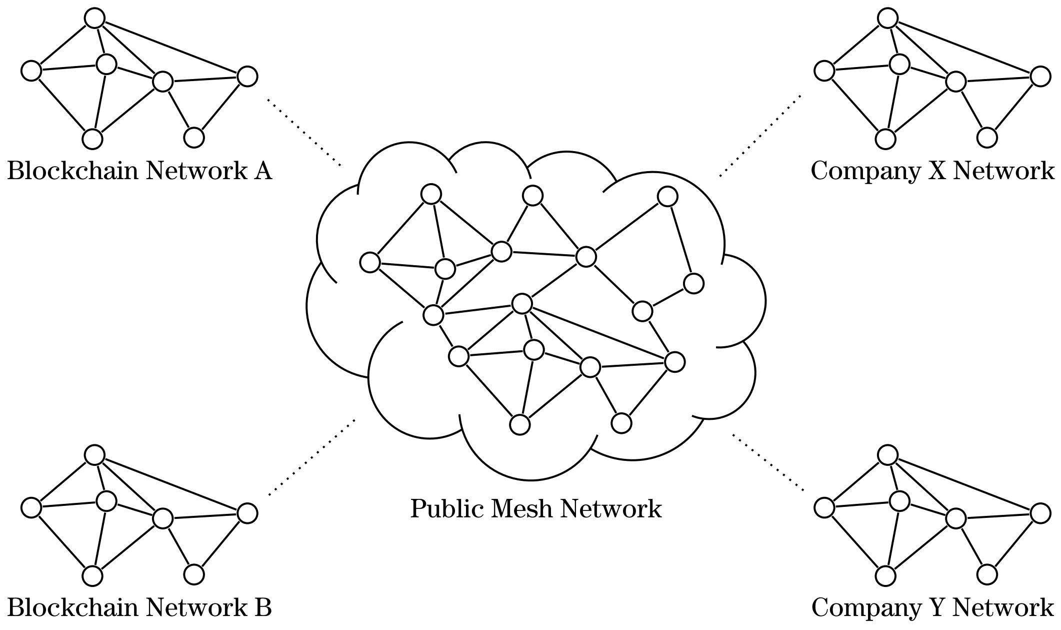

Figure 3: The Marconi Network is a network of networks

Individuals, network operators, and internet service providers can participate by contributing their bandwidth or compute resources to the network. In return for contributing resources and processing network traffic, nodes periodically receive network tokens known as marcos. The marco is the base unit of measurement for distributed networking and computing, the fuel consumed for network usage, administration, and smart contract processing.

While the Marconi Network can interoperate with existing internet infrastructure, it is also self-sustaining, capable of obviating existing network infrastructure by forming direct peer-to-peer connections to facilitate mesh networks that remove the need for hardware switches, routers, and bridges. In essence, the Marconi Network enables and incentivizes users to assemble and securely exchange network infrastructure resources without the physical, financial, and regulatory limitations that hinder traditional approaches to building, connecting, operating, and maintaining network infrastructure at scale. End users can utilize the network to access the internet or nearby compute power, either by procuring marcos or by mining them through operating a contributing node. Developers can utilize the network to create and deploy intelligent, decentralized networking applications that can be run by nodes or end users. Blockchain projects, private institutions, and enterprises can utilize the network and the platform it's built on to manage their infrastructure and develop smart distributed networking and cybersecurity services.

## 3 Design

In the next several sections we describe in detail the architecture of Marconi, including its network protocol, blockchain protocol, decentralized network administration, and token.

### 3.1 Network Protocol

The Marconi Network Protocol consists of three major components designed to allow secure, seamless connectivity and cooperation between network peers. These components are Marconi Pipe, smart packet contracts, and Marconi Link.

#### 3.1.1 Marconi Pipe

Marconi Pipe, or mPipe, is our implementation of a virtualized data link layer which provides a communication channel, or pipe, for transporting network traffic between peers. These pipes are a fundamental building block of the Marconi Network, and because they are established all the way down to layer 2 of the OSI model (see Figure 4), they allow several important capabilities such as custom packet routing and processing, increased security via packet-level encryption, and easy discovery of neighboring peers transmitting on the same local medium.

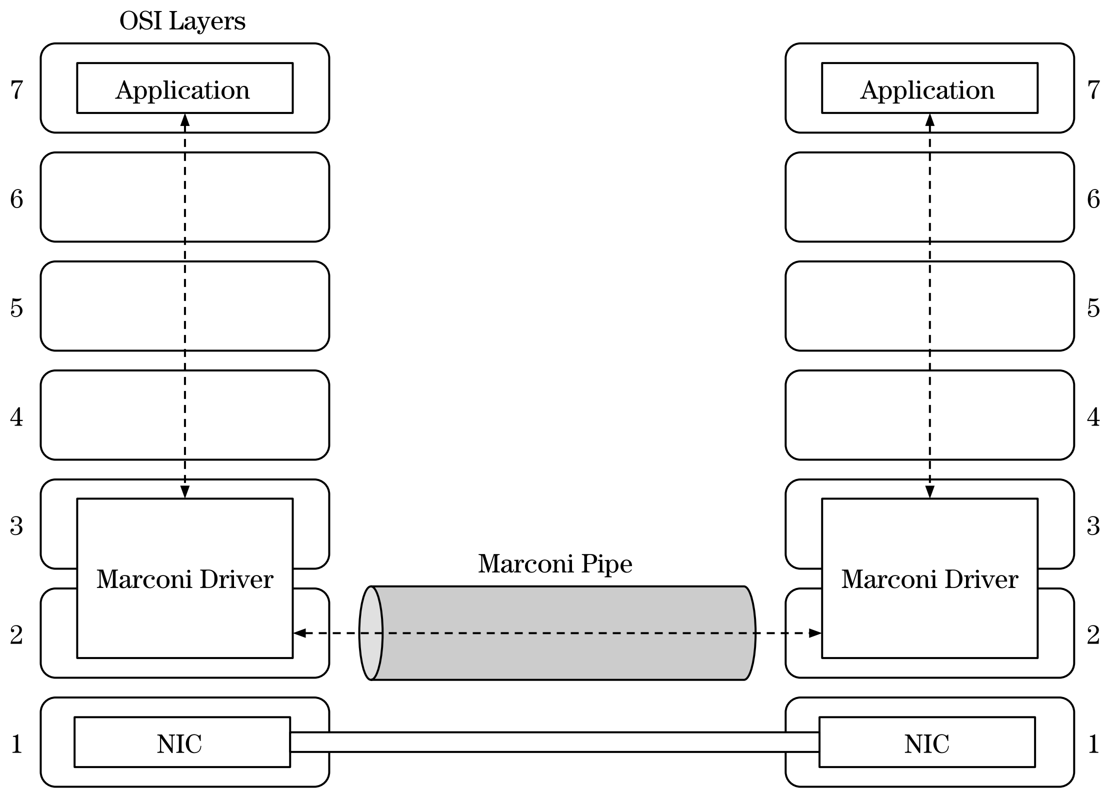

Figure 4: Marconi Pipe within OSI Model

When creating a pipe, a secure connection is formed between two peers by using a Diffie-Hellman exchange

[9] to create three shared secrets: one for data encryption, one for checksums to achieve data integrity, and one used as a seed. Each peer combines this seed with the current time truncated to a pre-defined granularity, say one minute, to obtain a new seed which changes over time. This in turn is used to mutate the data encryption secret and data integrity secret based on the current time interval (see Figure 5), similar to a time-based one-time password (TOTP), to help harden the data stream against attacks such as traffic pattern analysis.

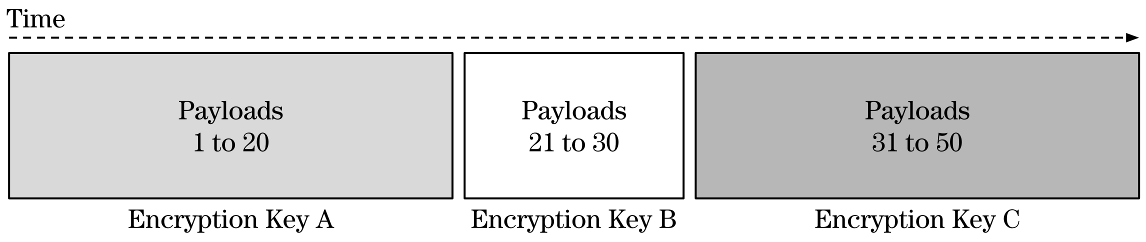

Figure 5: Mutating Encryption

The reason we use a system of symmetric keys is for performance. Packets will constantly be traversing many pipes, and useful cryptographic operations as defined in the Advanced Encryption Standard (AES) are directly supported in the instruction sets of common hardware.

Because mPipe is operating at the level of a network driver, it can be very performant both in terms of the aforementioned encryption and decryption as well as in terms of packet-level processing. This enables several interesting network features:

- **Packet Relay.** Within the network, packets can be relayed through multiple hops before exiting into the internet or private networks. Similar to onion routing, this improves privacy and helps protect against snooping, as only the network edge node is aware of the end user.
- **Packet Load Balancing.** Because connections are at OSI layer 2, the network has access to maximal information, therefore can easily load balance or even throttle network packets if necessary using a variety of different strategies to meet network requirements.
- **Packet Inspection.** Again because the network can access data from all the above OSI layers, it can make decisions based on payload contents. Even if data is encrypted at higher layers, interesting metadata is still available such as domain names and header information.

### 3.1.2 Smart Packet Contracts

With smart packet contracts, developers have the ability to run smart contracts against network packets to do smart routing and packet processing (see Figure 6). The Marconi Network provides a platform where developers can create decentralized networking applications using smart packet contracts.

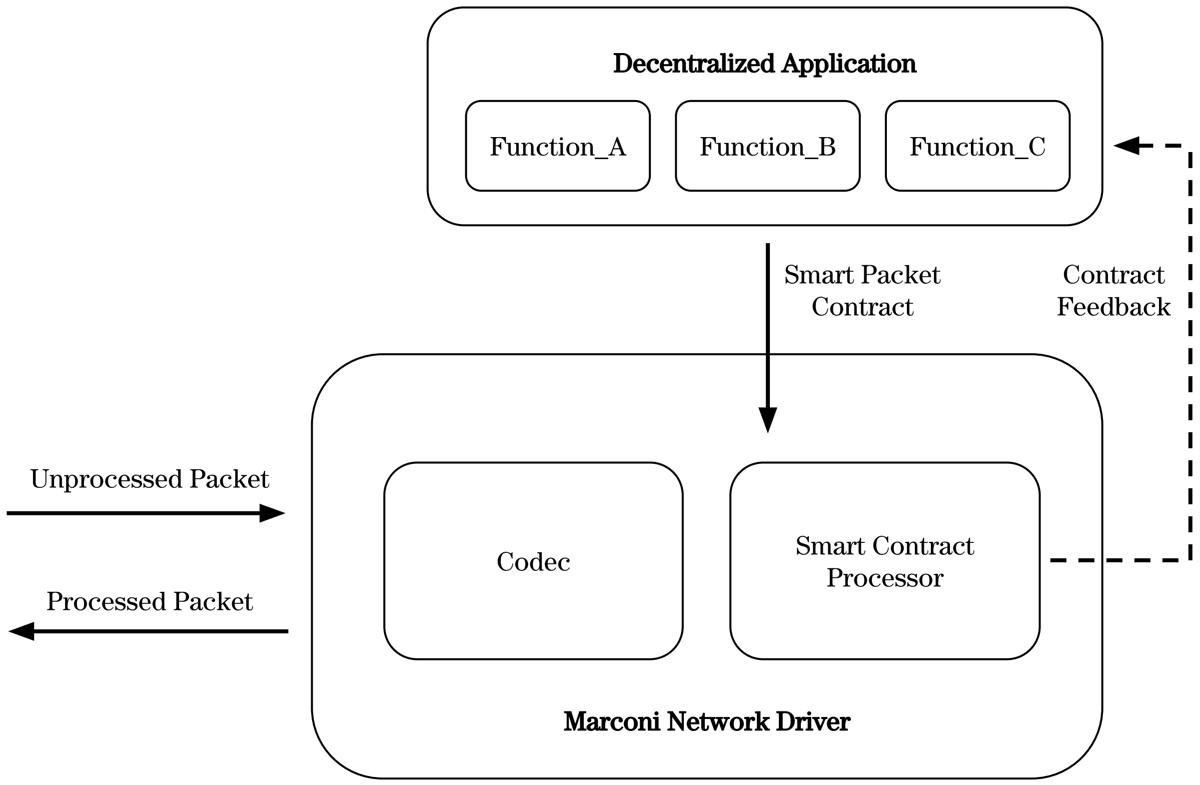

Figure 6: Packet Processing with Smart Packet Contracts

Example applications include software-defined networking (SDN), intrusion detection and prevention systems (IDS/IPS), anti-malware and anti-virus protection, content delivery networks (CDN), virtual private networks (VPN), and new blockchain protocols. These are discussed further in Section 4.

Applications are written in Marconi Script, a Turing-complete language which has access to network packets and compiles down to bytecode. We also provide three tiers of APIs in the Marconi Network Library: the Control API for operations that affect routing and traffic flow, the Content API for operations like inspecting payloads, and the Intelligence API for pattern analysis and machine learning. Once written, developers deploy applications to a global blockchain, the structure of which we discuss later in this paper. See Table 1 for more details about the Control API, and Figure 7 for how the API can be used.

| Function | Description |
|---|---|
| `mOpen` | Initiate opening an mPipe connection with a network peer. Accepts public key hash of desired network peer as input. |
| `mClose` | Terminate an mPipe connection. Accepts public key hash of desired network peer as input. |
| `mApply` | Apply rules to incoming and outgoing network packets. Accepts an mPipe ID and list of mFunc as input. Each mFunc is written using a language designed to easily allow passing functions as arguments in order to match contextual conditions and specify customizable actions on network packets. |
| `mList` | Returns the current states of mPipes and their applied mFunc list. |

Table 1: Smart Packet Control API

```
Contract PhishCatcher {
  Init(Address client_address) {
    TunnelRef handle = mOpen(client_address);
    mApply(handle, {PhishFunc});
  }
}

Status PhishFunc(PacketRef packet) {
  if (packet.src().url().as_string().match('[^\u0000-\u007F]')) {
    return Status::UNSAFE;
  }
  return Status::OK;
}
```

Figure 7: Example usage of Control API to detect suspicious URLs

The reader may be concerned about performance since applications may be performing low-level packet processing on a large amount of network traffic through a virtual machine layer. Fortunately, heavyweight processing such as pattern analysis can be done off the critical path by cloning and batching up traffic. For high-throughput real-time processing, we envision acceleration through custom hardware as the long term solution.

Another important goal which influenced our smart packet contract design is to be compatible with existing open source projects in this space, for example Snort, a popular network intrusion detection system and intrusion prevention system. We want to support these projects as well as make sure the smart packet contract development model is easily accessible to developers in these open source communities.

### 3.1.3 Marconi Link

Marconi Link, or mLink, is our solution for wireless communication. These links allow secure wireless transmission directly between network nodes, again designed down to layer 2 of the OSI model (see Figure 8) which provides access to network packets and thus enables smart packet contracts.

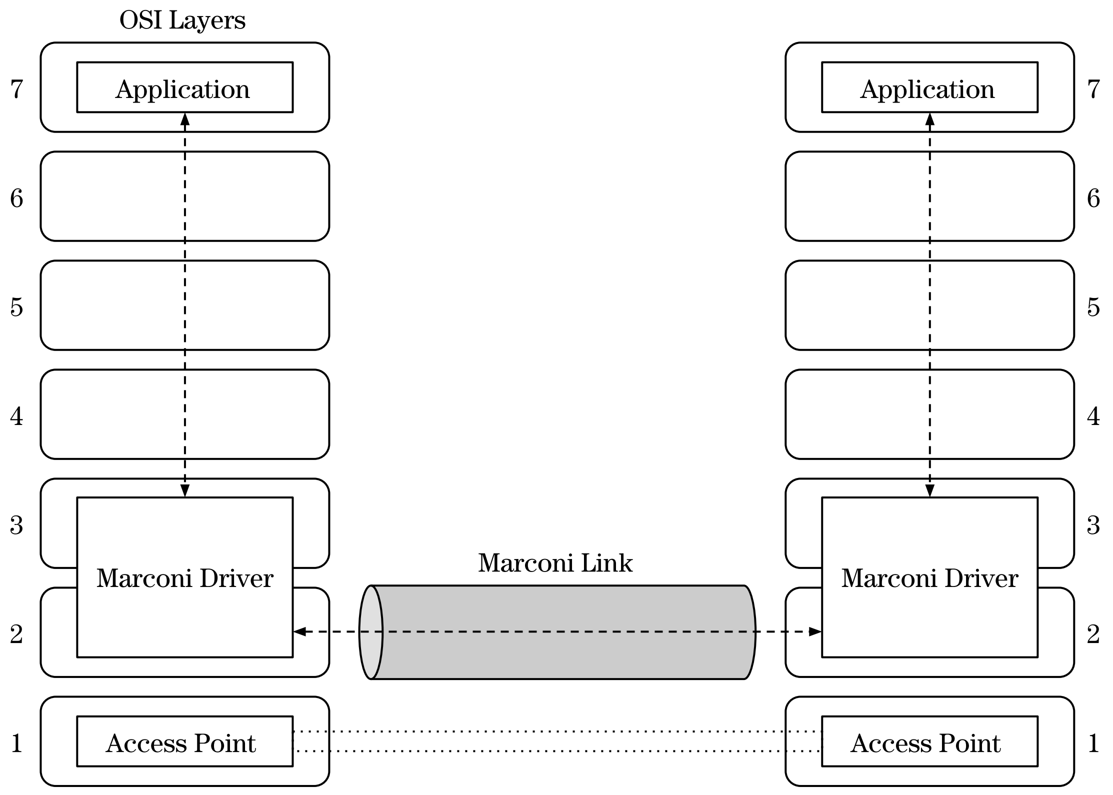

Figure 8: Marconi Link within OSI Model

Marconi Links are also a core building block for mesh networks which address problems like wireless network proliferation. The reader has probably experienced such a situation, for example being bombarded by twenty different network names when trying to connect to the internet over Wi-Fi. Through mLinks, nodes have the ability to easily to join the same network and are incentivized to do so, that way nodes can coordinate to transmit data more efficiently, which additionally yields better network range.

A node joins the mesh network through a discovery process in which the node first observes decibel levels to choose the best peers. The node then forms mLinks with those peers on the least loaded available channel. This involves a Diffie-Hellman exchange similar to mPipes in order to secure communication between nodes.

Marconi Links utilize the Marconi Wireless Driver, which is compatible with low-cost hardware repeaters and existing wireless standards such as Bluetooth and Wi-Fi, so no custom equipment is required to attain ranges on the order of 10 to 100 meters per device. And those willing to go the custom equipment route can take advantage of the relatively new U-NII-3 radio band to send information over multiple kilometers each hop, similar to wireless internet service providers. The Marconi Wireless Driver also gracefully handles wireless network congestion through channel negotiation for optimum connections between nodes.

#### 3.1.4 Designing Down to OSI Layer 2

When building network applications, developers sometimes take for granted today's internet infrastructure and topology. For instance, common approaches often make several assumptions such as a constant connection to the internet and an IP address obtained through the Dynamic Host Configuration Protocol (DHCP). But when operating outside of or slightly beyond the edges of the public internet, these assumptions don't always apply, for example in wireless peer-to-peer networks or when bridging between private networks. In these cases, OSI layer 2 must be considered, for instance in peer discovery and mechanisms like the Address Resolution Protocol (ARP). Additionally, to fully realize the long term vision of a truly decentralized network infrastructure where each node can act as a switch, router, or bridge, designing at layer 2 becomes a critical requirement.

Because components like Marconi Pipe and Marconi Link have access to OSI layer 2 data, they can observe MAC addresses to better understand physical network topology. This enables custom routing techniques for more performant traffic routing in some situations. For example in Figure 9, if node $A$ wants to send data to node $G$, it can utilize multiple outgoing links through nodes $B$, $C$, and $D$ in order maximize data throughput and redundancy. This is especially useful in the context of wireless mesh networks, in which physical links change fairly often as users move around, and which consist of heterogeneous devices with different connectivity capabilities such as smartphones, wireless access points, and wireless repeaters. This ability to perform low-level custom routing is also useful in the context of large wired networks such as private corporate networks, where alternative path optimization strategies can be preferable such as those used in the Open Shortest Path First (OSPF) protocol.

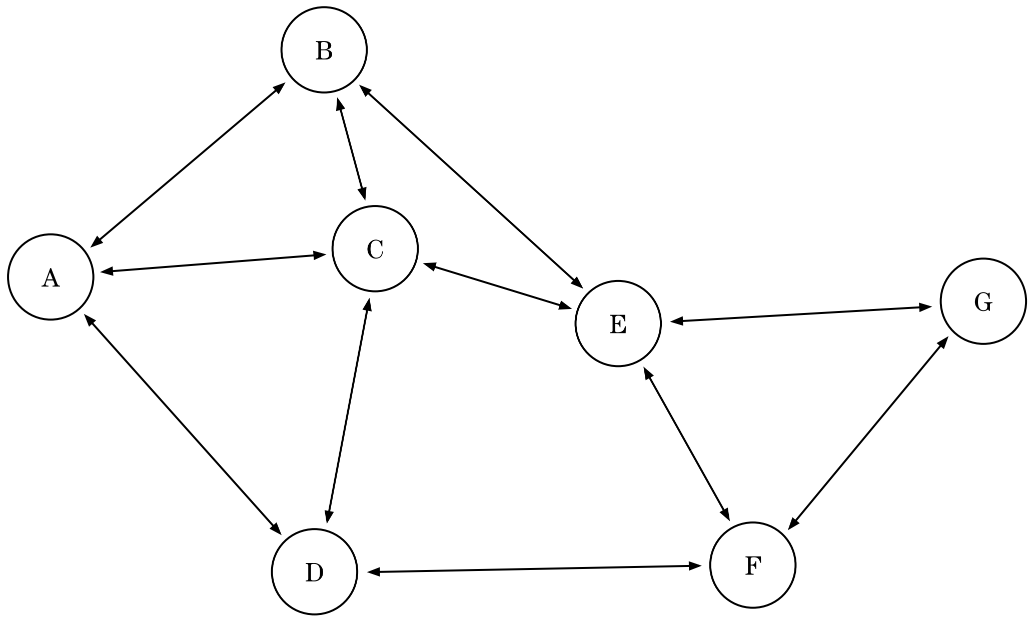

Figure 9: Marconi nodes and their physical links

### 3.1.5 Interoperability with Internet and IP Routing

To ensure the Marconi Protocol reaches wide adoption and fully realizes a decentralized network infrastructure, it needs to work with today's internet which is primarily powered by IP-based routing. To achieve this compatibility, when the Marconi Protocol interfaces with the internet, Marconi Pipes are virtualized as an overlay on top of IP-based connections as shown in Figure 10 by using standard IP and UDP headers.

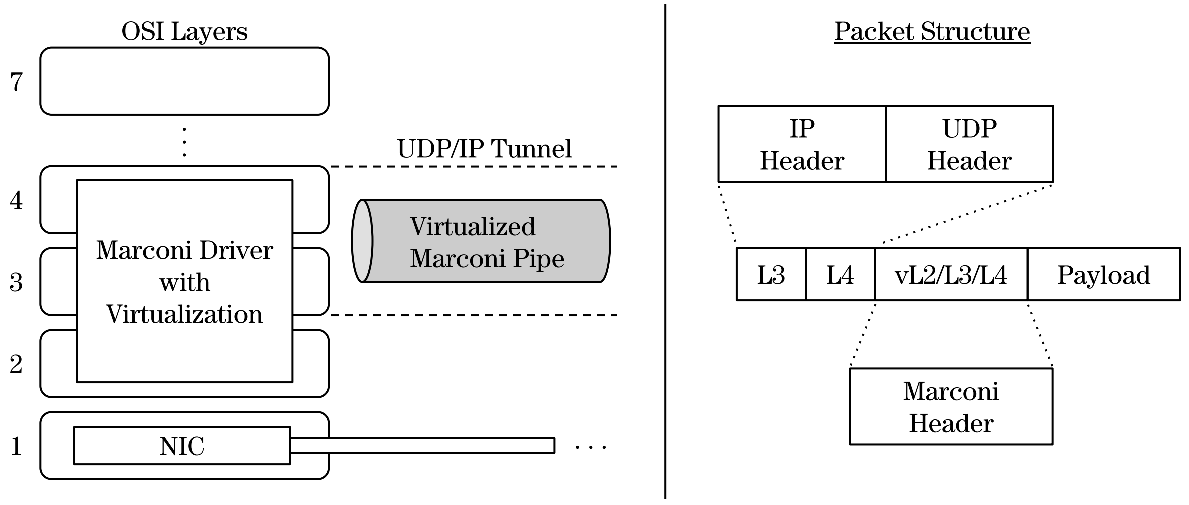

Figure 10: Virtualized Marconi Pipe and its packet structure

Conceptually this is similar to the Layer 2 Tunneling Protocol (L2TP) which is commonly used in VPN solutions. We additionally add an inner Marconi header which carries customizable metadata and control fields that get used upon packet arrival at the destination Marconi node. These virtualized Marconi Pipes still provide security benefits and smart packet contract processing, as well as easily integrate with our OSI layer 2 design for wireless peer-to-peer networks and bridged private networks. This overlay augments the existing internet with additional privacy, security, and flexibility.

For example, in order to increase privacy and security, nodes in the Marconi Network can use enhanced multi-layer encryption. This involves encrypting the network packet with a key previously negotiated with the intended destination, then adding successive layers of encryption using keys negotiated with each intermediate node via hops through prior nodes. This provides two benefits. First, only the intended destination can decrypt and read the final payload. Second, traffic obfuscation can be achieved by leveraging several intermediate nodes. With this scheme, because the packet was specially prepared by the original sender, each intermediate node only knows the immediately prior source of a packet and immediately subsequent destination, but nothing else about the full routing path due to nested layers of encryption which get iteratively peeled away at each hop. This is also known as onion routing.

## 3.2 Blockchain Protocol

The Marconi Network makes use of several blockchains for network formation, administration, and metering. Network formation and administration involve keeping track of nodes as they join or leave networks, while metering involves measuring and recording the bandwidth capacity and contribution of each node. We do this with a system comprised of a single global blockchain and many branch chains (see Figure 11). Branch chains always originate from the global chain, first splitting off and then periodically syncing back before rebranching. Mesh chains are a special type of branch chain which are used specifically for the purpose of organizing mesh networks.

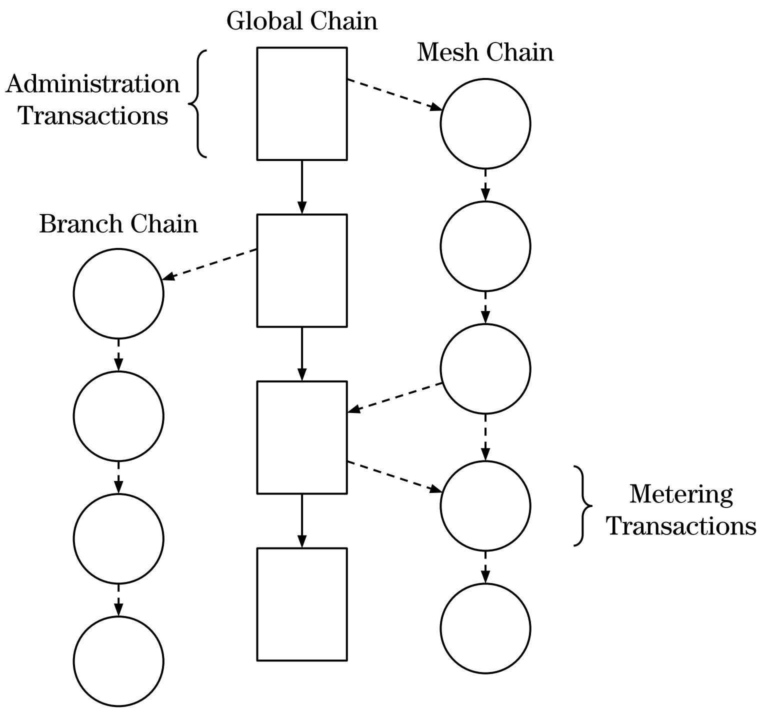

Figure 11: Global chain with branch chains

### 3.2.1 Global Chain

The global chain uses memory-hard proof of work to add blocks, and all nodes in the Marconi Network can contribute to this chain. The reason we use memory-hard proof of work is for ASIC resistance and so that the existing cryptocurrency mining community can begin mining without changing any of their hardware setup. The particular choice of hashing algorithm is decoupled from the rest of the system design and instead isolated as an implementation detail, that way the specific algorithm can be changed relatively easily if necessary, for example if an otherwise-viable ASIC is developed. The contents of the global chain include registration transactions containing nodes' public key identifiers, marco transactions, smart packet contract transactions, and checkpointing transactions which allow branch chains to periodically sync their aggregated state to the global chain.

### 3.2.2 Branch Chain

Branch chains are a generic construction that allow programmatically creating new blockchains that are attached to and run in parallel with the global chain. Each branch chain can have its own custom rules which are specified via branch contracts. This makes branch chains a flexible primitive that can be used in a variety of interesting ways.

One such example is a mesh chain, a type of branch chain which corresponds to a group of nodes that have decided to band together to form a mesh network. These nodes are typically but not always geographically near one another. Hence there can be many different mesh chains, and they come in two types: public, such that any node can contribute, and private, e.g. for enterprise networks. The purpose of mesh chains is to decouple mesh networks from the global chain in order to improve network robustness and scalability.

Mesh chains offload data and communication overhead from the global chain and allow nodes in mesh networks to often avoid interacting with the global chain. The contents of a mesh chain include registration transactions from nodes joining the mesh network as well as many proof-of-network-elements transactions. These are cryptographically verifiable exercises that are being repeated constantly by nodes in a mesh network, exercises which involve sending and receiving nonces to and from neighboring nodes to prove availability and bandwidth. We'll discuss proof of network elements later in more detail, but for now just know that nodes are incentivized to keep performing this exercise. Finally, rather than proof of work, mesh chains use proof of stake to add blocks, where network nodes who have participated in a larger number of proofs of network elements have a larger stake. The reason we chose proof of stake for mesh chains is for increased transaction throughput and to lower compute power requirements in contributing service nodes.

In order to join a mesh chain, a node must first have registered its public key and public key hash on the global chain. This allows the node to participate in smart contracts and marco transfers. Then the node needs to find out which mesh network to join. This is done by broadcasting to its neighbors to obtain their IP addresses, and by querying a distributed hash table (DHT) maintained by all network nodes which maps mesh network IDs to a list of IP addresses within each mesh network. Given this information, the node can register itself on the local mesh chain, a transaction that requires signatures from nearby peers which indicate they can reach the new node.

### 3.2.3 Proof of Network Elements

Our goal here is to design a decentralized metering protocol that captures network node availability and bandwidth over time, and incentivizes network bootstrapping and continued network participation. The main construction we employ is called proof of network elements, an exercise that any three nodes in the same mesh network can agree to participate in, for which they'll receive marcos upon each successful completion.

Some node $A$ initiates proof of network elements by first broadcasting to its neighbors and measuring the latency of their responses. Since lower latency will generally lead to accumulating more marcos, the node chooses the two closest neighbors $B$ and $C$ as peers for proof of network elements, and creates a transaction on the mesh chain establishing a smart contract which lists the public key hashes of these desired peer nodes. Each peer indicates its agreement to participate in proof of network elements by creating its own transaction to add its signature to the same smart contract. Once agreement occurs, the nodes form mPipes between one another.

Node $A$ then generates a nonce $n$ and creates a transaction which adds a hash of $n$ to a new smart contract. As seen in Figure 12, node $A$ then encrypts $n$ with its private key and sends the result, $Encrypt_A(n)$, to node $B$. This receiving peer can verify the message by decrypting with $A$'s public key. Then $B$ repeats the process, yielding a message $Encrypt_B(Encrypt_A(n))$ which it sends to $C$. Node $C$ does the same, sending the payload $Encrypt_C(Encrypt_B(Encrypt_A(n)))$ back to $A$. Finally, $A$ commits this payload to the mesh chain in a transaction that updates the existing smart contract. This payload can be verified by any node in the Marconi Network by using the public keys belonging to $A$, $B$, and $C$.

Note that node $A$ can simultaneously participate in proof of network elements in the opposite direction, by starting a separate smart contract and sending an encrypted nonce to $C$ who sends to $B$ who sends back to $A$. This allows measuring links bidirectionally, since there may be differences in bandwidth or latency in the real world depending on the direction of data flow.

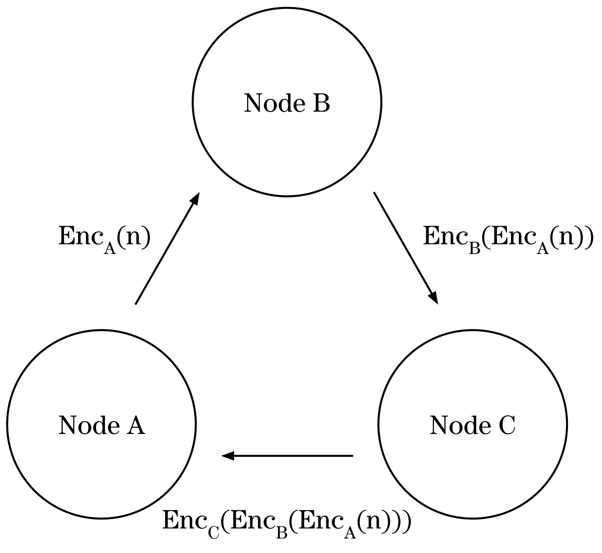

Figure 12: Nodes performing proof of network elements by encrypting a nonce $n$

To demonstrate higher bandwidth, nodes can use larger nonces if their peers agree to it. Since nodes can keep repeating proof of network elements to keep gaining marcos, and since these completed exercises are all encoded in the mesh chain, over time they present a reasonably accurate view of bandwidth and availability for each node. Furthermore, the results of proof-of-network-elements exercises can be uploaded to the mesh chain in delayed fashion if necessary, since it's not of critical importance whether uploading occurs right away or several blocks later. The data can still be easily verified in either case. Because of this asynchronous property, the participating nodes aren't blocked on waiting for mesh chain transactions, hence can continue making progress independently by sending more nonces to their peers. This asynchronous property leads to large overall data throughput among peer nodes and in mesh chains.

#### 3.2.4 Chain Optimizations

While mesh chains help offload data from the global chain, they themselves can potentially grow large due to proof of network elements. To mitigate this, we employ several techniques such as truncating mesh chain data older than a few months (recall that aggregated mesh chain state is periodically synced to the global chain), and sampling methods where nodes participate in proof of network elements a small fraction of overall time. Note that these truncation and sampling parameters are actually tunable in the case of private mesh chains, so for instance a chain creator could choose to keep a longer or shorter history of proof-of-network-elements data.

We're also investigating ways to decrease the global chain storage size, for example by submitting data related to smart packet contracts to mesh chains where possible, rather than to the global chain. Another area of investigation involves custom compression schemes for saving transmission bandwidth as well as persisted chain size.

Additionally, we're experimenting with classifying and hashing sets of transactions in different ways, such as grouping simple value transfers separately from smart packet contract transactions, so that nodes can specialize in certain functionality and therefore avoid syncing the entirety of global chain data, and instead only sync certain classes or subsets. This can significantly improve block sync time, both when bringing up a brand new node as well as in steady-state forward-sync mode.

From combining these optimization strategies, early estimates show potential savings of between 30 and 50 percent of global chain size.

Finally, related to chain size optimizations, we think we can also improve the physical layout of persisted chain data and the way it gets read from and written to, for instance with smarter IO buffering. This will improve the performance of traditional hard disk drives which in turn will lower the cost of service nodes. This is because operators won't be forced to use solid-state drives which, for example, are currently a common work-around for getting reasonable performance from Ethereum nodes.

## 3.3 Network Administration

### 3.3.1 Network Actors

There are generally three types of actors who participate in the Marconi Network. First is the node operator, who runs one or several service nodes that contribute resources to a public mesh network. These nodes perform functions chosen by the operator, which might include allowing end users to connect, processing network traffic, confirming blocks on the global chain or mesh chain, traffic metering, or bridging different mesh subnetworks. The operator may choose to contribute to an existing mesh network, or generate a new mesh network as a branch from the global blockchain by invoking the proper branch contract for creating a new mesh chain.

Second is the network operator, who needs to coordinate a fleet of nodes in particular ways. The network operator invokes a branch contract on the global chain which generates a new branch chain, for example to form a new private mesh network (see Figure 13). Also specified in this branch contract are the rules specific to the branch chain such as which smart packet contracts are enabled, e.g. for network efficiency through software-defined networking, or for end user protection through anti-phishing tools. Examples of network operators include companies, local ISPs, or new blockchain protocols creating their own token economy.

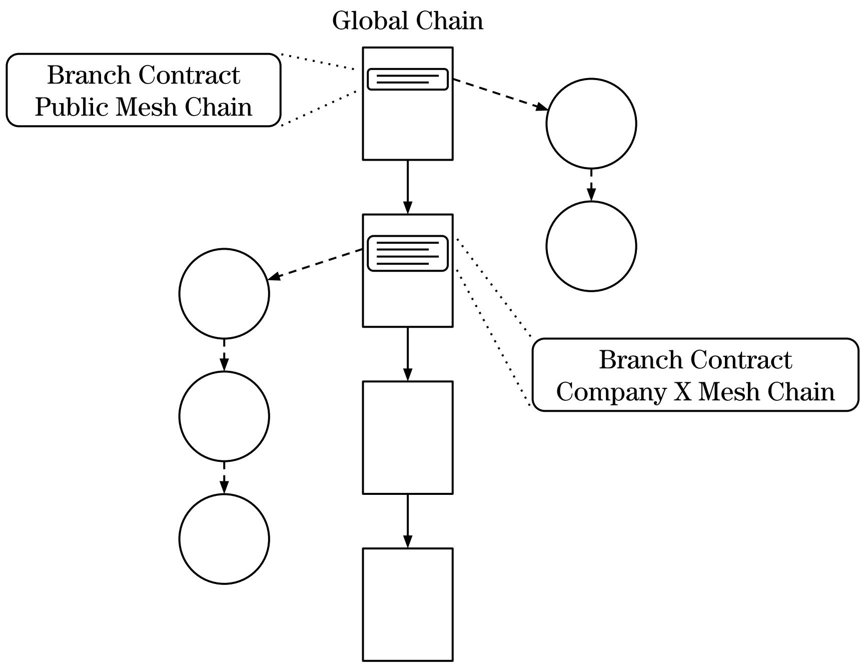
Global Chain
Branch Contract Public Mesh Chain
Branch Contract Company X Mesh Chain

Figure 13: Public and private mesh chains created using branch contracts

Third is the end user, who may join a public mesh network at any time or a private mesh network if authorized to do so. When joining a public mesh, the user opts into the smart packet contracts they choose, while in a private mesh they must abide by the rules of that mesh.

### 3.3.2 Network Formation

As an example, let's examine the case of a node operator who wants to bring up a new service node on an existing mesh chain. The new service node joins the Marconi Network through the following sequence of steps:

1. **Bootstrapping.** The new service node contacts a small set of bootstrap nodes. These bootstrap nodes keep track of a subset of active nodes in the Marconi Network that store copies of global chain data, and respond to the new service node with metadata about these nodes. The new service node then queries a random sample of nodes from this subset to observe the latency of their responses. From each of the few nodes that responded quickest, the new service node requests a list of the node's peers. The new service node then repeats this process of querying a sample of nodes in these lists, measuring their latency, and obtaining lists of their peers, until it has identified a set of close by neighbors that it will use as global chain peers for syncing global chain data.
2. **Synchronization.** The new service node syncs historical blocks from the global chain by downloading data from its peers, and also begins listening to new blocks in forward-sync mode.
3. **Registration.** The new service node broadcasts a transaction to its peers which registers its public key identifier on the global chain.
4. **Discovery.** The new service node queries its global chain peers for metadata about potential mesh chain peers. Similar to the iterative process of finding peers in the bootstrapping step, the new service node narrows the list of potential mesh chain peers by observing latency and bandwidth of requests and responses to and from those peers. Once the most optimal peers are identified, the new node begins syncing mesh chain blocks from peers in the respective mesh chain.
5. **Negotiation.** The new service node creates a transaction on its desired mesh chain establishing a smart contract which lists the public key hashes of its desired immediate peer nodes. Each peer indicates its agreement by creating its own transaction to add its signature to the same smart contract.
6. **Setup.** The new service node forms an mPipe with each of its immediate peers by using Diffie-Hellman exchanges to construct shared secrets.
7. **Maintenance.** The new service node performs proof of network elements with its immediate peers on an ongoing basis. Any sensitive communication occurs through the mPipes established in the previous step.

If a node is only mining the global chain, as opposed to contributing to a mesh chain, then the sequence of steps is simpler. In such a case, only the bootstrapping and synchronization steps are necessary.

### 3.3.3 PeerRank

The Marconi Network uses a reputation system to measure node and subnet quality. The network can track each node's capacity and availability over time by observing the results generated by its proof-of-network-elements exercises in order to produce a normalized score based on these metrics relative to nearby peers. Nodes that behave well and coordinate with peers who have a strong score will themselves tend to have a better score. This weighting system is called PeerRank and yields a stack rank of network nodes and subnets which can be used in a variety of situations such as determining which nodes can accept new connections from network peers and still maintain their current connections. Scores are also a convenient signal for end users when deciding which nodes to connect to.

The PeerRank score for each node is periodically synced to the global chain based on the node's recorded behavior in its respective mesh chain. This associates the score with a public key identifier so that any other node can observe and make use of it.

#### 3.3.4 Reducing Complexity in Mesh Networks

The Marconi Network Protocol supports wireless networks and peer-to-peer mesh networks. Managing connections between nodes and ensuring performant traffic routing in such networks can be challenging since the number of possible connections grows as $O(n^2)$ where $n$ is the number of nodes.

Consider for example a mesh network of just four nodes $A, B, C$, and $D$ (see Figure 14). If node $A$ wants to send data to node $D$, there are many possible paths to choose from: $AD, ABD, ACD, ABCD$, and $ACBD$.

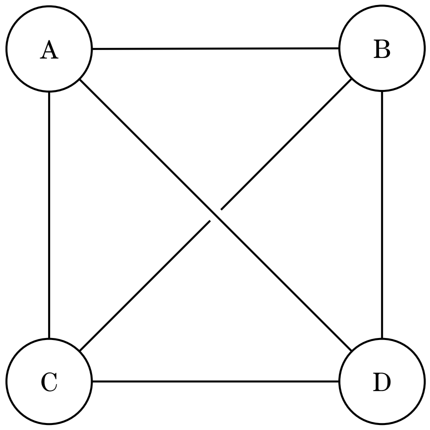

Figure 14: A mesh network of four nodes

To reduce this complexity, we combine and bind multiple OSI layer 2 connections in a given network node into a single virtual network interface with OSI layer 3 configuration data, similar to how operating systems can bridge multiple physical network interface controllers to achieve higher capacity and redundancy for network connectivity. This binding removes complexity for node and network operators as well as for developers writing smart packet applications. It also provides connection redundancy and fault tolerance in case a network link is lost, and increased throughput since it grants the ability to distribute outgoing traffic over multiple paths.

### 3.4 Token

The marco is the token that powers the Marconi Network. These utility tokens can be used for network bandwidth or consumed as fuel for processing network traffic and running smart contracts. Service nodes receive marcos when confirming blocks or successfully completing proofs of network elements. Marcos are also required for some types of network operations such as staking in a mesh chain or creating a transaction to start a new private mesh chain.

The number of marcos required to perform work in the Marconi Network is determined by several parameters. It can be modeled as a function $f(\alpha, \beta, \gamma, \delta)$ where $\alpha$ is the bandwidth used, $\beta$ represents the quality of service of the delivered bandwidth, $\gamma$ represents the complexity of any executed smart contracts in terms of compute and storage, and $\delta$ is the fuel price set by the client. A provider may accept the fuel price in which case the work is performed, or reject the fuel price in which case the work is ignored.

Similar to other minable fuel-based blockchains, the supply of marcos will initially increase at a larger rate to incentivize miners to bootstrap the network, then slow down to reach a minimum rate of increase after a total of twenty years (see Figure 15).

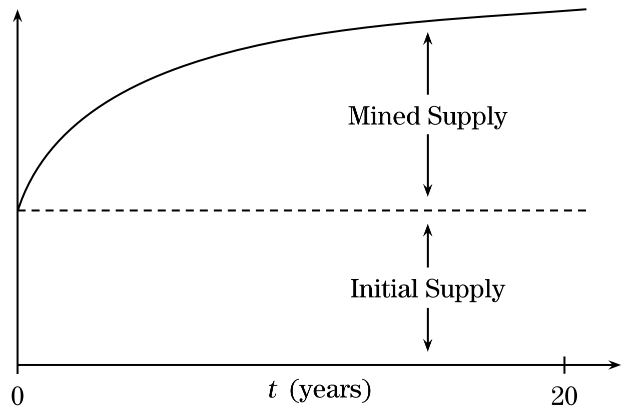

Figure 15: Token supply over time

# 4 Applications

The Marconi Network and its smart packet contracts provide a robust decentralized network infrastructure and platform that enables developers to build powerful applications. In this section we'll discuss a few demonstrative examples. We also believe that ultimately the developer community will imagine and create many additional compelling applications.

## 4.1 Blockchain Protocols

### 4.1.1 Decentralized Network Infrastructure

New blockchain protocols can be developed and launched on top of the Marconi Platform using branch contracts and branch chains, gaining its built-in benefits for free such as security and net neutrality. And with no code changes, existing blockchain protocols can begin using the Marconi Network for inter-node communication, taking advantage of its augmented and eventually fully decentralized network infrastructure.

### 4.1.2 Token to Blockchain Migration

Another interesting application related to blockchain protocols is the ability to migrate token implementations to their own blockchain. For example, if a token is implemented in terms of a common standard such as ERC20, then its rules can easily be specified as the basis of a branch contract creating a new branch chain that originates from the global chain, allowing the token to be migrated to its own separate chain running in the Marconi Network. Account balance state can be aggregated into a genesis block for this new branch chain to break dependencies on any previous chain data such as Ethereum's gargantuan data directory. Submitting transactions to this new chain allows transferring tokens between accounts since the transactions can invoke functions in the originating branch contract on the global chain. This new chain still periodically checkpoints aggregated state back to the global chain, for example a snapshot of account balances gets written into the originating branch contract to reflect what transpires in the branch chain.

## 4.2 Security Applications

### 4.2.1 Anti-Phishing, Anti-Malware, and Anti-Virus Protection

This protection can be provided out of the box for end users and even for corporate networks without installing any special software or hardware by instead using smart packet contracts. This is possible through analyzing packet payloads if unencrypted or otherwise through analyzing packet headers, size of packets, number of packets, and domain names.

For instance a common phishing strategy involves constructing misleading URLs which look similar to URLs that users trust, e.g. replacing an ASCII character with a nearly identical unicode character, then registering the resulting domain name and disguising the new site to appear as the trusted site. Detecting such misleading URLs is quite simple with a few lines of smart packet contract code (recall Figure 7). While some of the most popular browsers and email clients already implement anti-phishing protection, it remains an afterthought or completely absent from many of the less popular or more basic user applications such as text messaging. By solving this problem just once on the Marconi Platform at low-level networking layers, it's no longer necessary to reimplement the same solution multiple times at higher layers in every possible user program. And the Marconi implementation can be extended with other common anti-phishing techniques such as a database of known bad sites and content, or a machine learning model trained on such sites and content. Finally, the Marconi Platform allows developers to require marcos for usage of their decentralized application if they so choose, marcos which could in turn be used to incentivize end users to report suspicious sites or content.

#### 4.2.2 Intrusion Detection and Prevention System (IDS/IPS)

In private networks, monitoring can be put in place to detect anomalies in traffic and malicious activity happening on the network, protecting from both external and internal attackers by either notifying administrators or actively blocking suspicious packets.

#### 4.2.3 Virtual Private Network (VPN or dVPN)

A private network can be extended over a public or shared network, preventing unauthorized access via authentication and protecting privacy via encrypted traffic and a level of indirection. Marconi Link is especially helpful here, for users who demand strong obfuscation of network routes, since wireless hops are much more difficult to track than hops through wired connections. And in both the wired and wireless case, users enjoy the benefits of extra security with low-level encryption at OSI layer 2. Users can even rotate among multiple entry nodes and exit nodes to further obscure their traffic.

### 4.3 Networking Applications

#### 4.3.1 Secure Field Network

Using the same technology that supports the Marconi Protocol, secure field networks can be rapidly deployed in battlefield and disaster relief situations with their traffic history stored on a ledger that can be audited after the network is decommissioned.

#### 4.3.2 Internet-of-Things (IoT) Device Management

As the number and capabilities of IoT devices increase, so too will the number of possible applications they can power, as well as the need for secure communication between them. Through Marconi and its smart packet contracts, these devices can securely and gracefully interoperate and be coordinated on-demand to work together to solve complex problems, putting locally available hardware to good use in a modern day variant of distributed grid computing. Taken together, the Marconi Network and APIs provide a platform that can put directly into the hands of developers the combined power of multiple smartphones, workstations, and even GPUs in self-driving cars to benefit users both nearby and far away, enabling a new class of applications we've never before seen.

#### 4.3.3 Content Delivery Network (CDN or dCDN)

Large or commonly accessed internet content can be cached by nodes in the Marconi Network so that it can be served more locally to consumers, yielding more efficient use of bandwidth and improved latency. Examples of such content include video files or historical blocks from various popular blockchains which must be synced by new mining nodes. This would drastically decrease the time needed to set up mining nodes from scratch.

A decentralized, peer-to-peer CDN is also a competitive option for content providers because there's no overhead cost for initially bringing up the delivery network, nor any ongoing cost for maintaining network hardware. There's only the cost of deploying and serving the bytes themselves, and the ability to do traffic metering is already built into the Marconi Network.

#### 4.3.4 Software-Defined Networking (SDN)

Machines can be managed and controlled through distributed network virtualization which allows dynamically reconfiguring and reprogramming network hardware from a single console, for example spinning up new nodes for load balancing or traffic shaping.

# 5 Security

In this section we discuss in detail several types of attacks to which networks and blockchain systems are potentially susceptible, and how the Marconi Protocol is designed to mitigate these attack vectors. Because Marconi addresses these attacks at a foundational layer, new blockchain protocols that build on top of Marconi also gain the same protection.

## 5.1 Transaction Flooding Attack

A transaction flooding attack might add many no-op or otherwise useless transactions onto the global chain or a particular branch chain, in an attempt to overwhelm the network with so much work that it prevents normal transactions from being added or processed. To mitigate this attack, most of the transaction types in the Marconi Protocol require a small security deposit which only gets refunded after some time has passed, and only if the operation completes successfully. For example, if an attacker tries to generate many branch chains in a short span of time, the number of chains will be limited by the attacker's available funds, since any transaction without the necessary backing deposit will be rejected immediately. And in these newly created branch chains, if there aren't any nodes registered and contributing to them—which itself requires a security deposit—then the deposits used to create the branch chains will be slashed rather than refunded.

## 5.2 Packet Injection

Injecting forged network packets into communication streams becomes much more difficult in the Marconi Network due to each hop between nodes being individually protected by OSI layer 2 encryption with a particular combination of symmetric keys via the mPipe or mLink implementation. Without the secret keys, an attacker won't be able to decrypt the data, which in turn makes it hard to analyze the traffic in order to figure out a possible point for packet injection. Nor will the attacker be able to properly encrypt any malicious packets they construct, hence target nodes can easily differentiate them from trustworthy packets. Even replaying previous packets becomes much less effective due to the mutating encryption keys explained in Section 3.1.1, since the same data may no longer be considered valid due to timestamp mismatch.

## 5.3 Sybil Attack

The first line of defense against Sybil nodes in the Marconi Network is cost. Because public key registration and most other transactions require a security deposit, the number of identities an attacker can construct is limited by their available funds.

The second line of defense against Sybil nodes is PeerRank. Honest nodes tend to prefer working and coordinating with nodes they can trust since they can obtain more marcos this way and avoid slashed deposits as well as avoid drops in their own PeerRank score. For instance, if a node fails to send or receive an appreciable fraction of nonces during a proof-of-network-elements exercise, not only will it and its peers receive less marcos, but their PeerRank scores will be adversely affected as well.

It is an explicit non-goal to protect the network against irrational behavior. For example, an operator could maintain a well-behaved node for an extended period of time in order to gain trust, then suddenly begin misbehaving. Such behavior would result in losing marcos, whereas the operator would have continued gaining marcos if they had kept up the node's good behavior. Therefore a rational actor isn't a threat to the network, but there's only so much that can be done against irrational actors.

### 5.4 Quantum Attack

Many widely-used cryptographic algorithms can be cracked by a quantum computer utilizing a large enough number of qubits. Though it's unlikely that such computers exist today, it's likely that they will exist in the near future. The developers of the Marconi Protocol are very much interested in hardening the system with post-quantum cryptographic techniques as they continue to unfold.

## 6 Conclusion

We have proposed a system allowing smart contracts for network packets as well as programmatically jump starting branch chains and secure networks. We began by considering the current state of network infrastructure and noticed it's insecure, difficult to manage, and centrally controlled. We also realized that this problematic infrastructure is being relied upon by many existing blockchain projects, by private and local networks, and even more generally by the internet itself. We remedied the situation with a much-needed revamp in the way we think about, deploy and use network infrastructure, by providing a network protocol designed down to OSI layer 2 with robust security, performance, and peer-to-peer meshing capabilities, and by providing a blockchain protocol to enable trustless interoperation and exchange of network resources and incentivize participation in bootstrapping this new network. Finally we empowered developers to build novel distributed and decentralized applications on top of this platform.

## 7 Future Work

In prior sections we have restricted our discussion mainly to protocol and software development work. But in addition to these areas, the Marconi Protocol also presents many interesting opportunities to investigate in the realm of hardware.

### 7.1 Ubiquitous Hardware

By design, the Marconi Protocol works with existing hardware owned by miners and casual users, but we imagine that some individuals may decide to specialize in a particular facet of network resources such as hashing power to add blocks to the global chain or NICs and access points for connectivity and bandwidth.

Experimenting with different hardware setups to maximize mining output is indeed an interesting avenue for future exploration. Depending on the geographical parameters, the mining setups could look very different. For example having access to cheap electricity would naturally point to using more GPUs, while being located in a dense urban environment with many smartphone users might incentivize focusing on connectivity hardware.

And to further expand and increase access to the Marconi Network, we're also excited by the idea of developing open source drivers for low-end devices and hardware such as the Raspberry Pi. This would allow node operators to build and extend their networks at very low cost as shown in Figure 16.

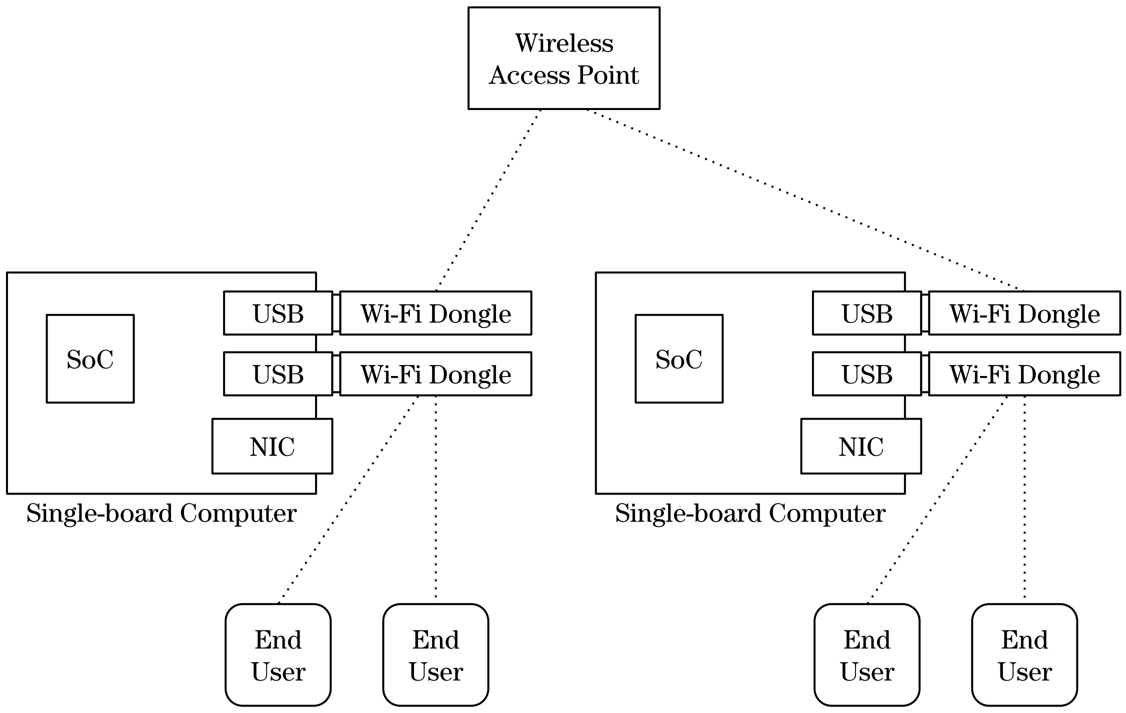

Figure 16: Extending a network with low cost hardware

## 7.2 Enterprise Hardware

Another promising area to investigate in the realm of hardware is the needs of private enterprises. It's likely worthwhile to tailor custom hardware configurations to optimize for client-specific workloads on private mesh chains that differ significantly from the public mesh and global chain cases.

An even larger benefit would come from designing and releasing open hardware specifications, tooling, and an open instruction set architecture that provide hardware acceleration for smart packet contracts. This way companies have the ability to relatively easily design, fabricate, and assemble custom hardware to lower costs as well as drastically increase speed and efficiency, thereby enabling a new set of advanced applications and functionality that were previously impossible.

See Figure 17 as an example of what an enterprise-grade hardware configuration might look like.

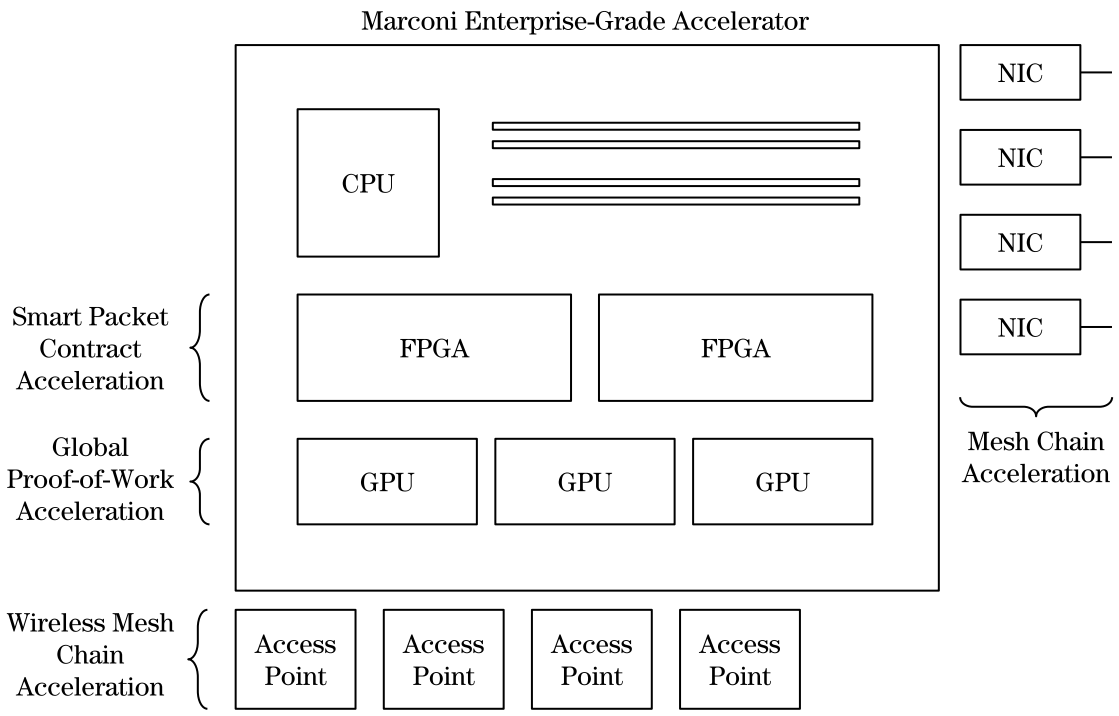
Marconi Enterprise-Grade Accelerator
CPU
NIC
NIC
NIC
NIC
Smart Packet Contract Acceleration
FPGA
FPGA
Mesh Chain Acceleration
Global Proof-of-Work Acceleration
GPU
GPU
GPU
Wireless Mesh Chain Acceleration
Access Point
Access Point
Access Point
Access Point

Figure 17: An enterprise-grade hardware configuration

## References

[1] Market capitalization and number of cryptocurrencies at https://coinmarketcap.com. Accessed September 2018.

[2] T. Kiravuo, M. Särelä, and J. Manner, "A Survey of Ethernet LAN Security," In *IEEE Communications Surveys & Tutorials*, volume 15, pages 1477-1491, 2013.

[3] C. Timberg, "Net of Insecurity: A Flaw in the Design," *The Washington Post*, May 30, 2015.

[4] Ethereum data directory size at http://bc.daniel.net.nz. Accessed February 2018.

[5] Cryptocurrency Exchange 24 Hour Volume Rankings at https://coinmarketcap.com/exchanges/volume/24-hour/. Accessed April 2018.

[6] M. Mulders, "Binance Hack Linked To Viacoin Pump (Official Update)," *Hacker Noon*, March 7, 2018. Available: https://hackernoon.com/alleged-hack-of-binance-linked-to-viacoin-pump-bb9066bf96bf.

[7] B. Kobb, "FCC Creates Unlicensed Data Radio Service," *Wired*, January 9, 1997.

[8] Federal Communications Commission (FCC), "Unlicensed National Information Infrastructure (U-NII) Devices in the 5 GHz Band," ET Docket No. 13-49, FCC 14-30, May 1, 2014.

[9] W. Diffie and M. Hellman, "New Directions in Cryptography," In *IEEE Transactions on Information Theory*, volume 22, number 6, pages 644-654, 1976.