RightMesh™

The **Power** of **Connectivity** in the Hands of the **People**

Decentralized Mobile Mesh Networking Platform Using Blockchain Technology and Tokenization.

rightmesh.io

TECHNICAL WHITE PAPER

December 14, 2017	3.1
DATE LAST UPDATED	VERSION

This is a DRAFT document for open community review. Subject to change.

This document constitutes a description of the RightMesh platform and the functionality of the RightMesh tokens; it is for informational purposes only and may change as the RightMesh technology develops over time.

Technical White Paper:
A Decentralized Mobile Mesh Networking Platform Powered by Blockchain Technology and Tokenization

Authors: Dr. Jason Ernst, Dr. Zehua (David) Wang, Saju Abraham, John Lyotier, Chris Jensen, Melissa Quinn, Aldrin D'Souza

Change Log

Date	Version	Comments/Edits	Contributor
Sept 7, 2017	v.1.0	Initial version of the technical paper	JE
Dec 4, 2017	V.2.0	Updated the Token Model, Added Networking Details, Added some basic assumption calculations	JE
Dec 10, 2017	V.2.0	Updated Legal Statements	JL SO
Dec 13, 2017	V.3.0	Updated legal statements and global edits	MQ JL
Dec 15, 2017	v.3.1	Minor Edits	JL

TABLE OF CONTENTS

THIS IS NOT A PROSPECTUS OF ANY SORT

***This document does not constitute a prospectus of any sort**; it is not a solicitation for investment and does not in any way pertain to an offering of securities in either Canada or the United States, and Canadian and United States residents are expressly excluded from contributing in exchange for any RightMesh Tokens in the public contribution offering. This document constitutes a description of the RightMesh platform and the functionality of the RightMesh tokens; it is for informational purposes only and may change as the RightMesh technology develops over time.*

***DISCLAIMER:** This draft RightMesh Technical White Paper is for information purposes only. Left of the Dot Media Inc. (dba Left), RightMesh GmbH (a Swiss Company in Incorporation), and all affiliated and related companies do not guarantee the accuracy of the conclusions reached in this paper, and the white paper is provided "as is" with no representations and warranties, express or implied, whatsoever, including, but not limited to: (i) warranties of merchantability, fitness for a particular purpose, title or noninfringement; (ii) that the contents of this white paper are free from error or suitable for any purpose; and (iii) that such contents will not infringe third-party rights. All warranties are expressly disclaimed. Left, RightMesh GmbH and its affiliates expressly disclaim all liability for and damages of any kind arising out of the use, reference to, or reliance on any information contained in this technical white paper, even if advised of the possibility of such damages. In no event will Left, RightMesh GmbH, or its affiliates be liable to any person or entity for any direct, indirect, special or consequential damages for the use of, reference to, or reliance on this white paper or any of the content contained herein.*

Recipients are specifically notified as follows:

- ***No offer of securities:** RightMesh Tokens (as described in this RightMesh Technical White Paper) is not intended to constitute securities in any jurisdiction. This White Paper does not constitute a prospectus nor offer document of any sort and is not intended to constitute an offer or solicitation of securities or any other investment or other product in any jurisdiction.*
- ***No advice:** This RightMesh Technical White Paper does not constitute advice to contribute in exchange for any RightMesh Tokens, nor should it be relied upon in connection with, any contract or contribution decision.*
- ***No representations:** No representations or warranties have been made to the recipient or its advisers as to the accuracy or completeness of the information, statements, opinions or matters (express or implied) arising out of, contained in or derived from this White Paper or any omission from this document or of any other written or oral information or opinions provided now or in the future to any interested party or their advisers. No representation or warranty is given as to the achievement or reasonableness of any plans, future projections or prospects and nothing in this document is or should be relied upon as a promise or representation as to the future. To the fullest extent, all liability for any loss or damage of whatsoever kind (whether foreseeable or not) which may arise from any person acting on any information and opinions contained in this RightMesh Technical White Paper or any information which is made available in connection with any further enquiries, notwithstanding any negligence, default or lack of care, is disclaimed.*

Risk warning: *Potential contributors should assess their own appetite for such risks independently and consult their advisors before making a decision to contribute in exchange for any RightMesh Tokens.*

Introduction

This technical paper is intended as a supplement to the *RightMesh White Paper* (*http://www.rightmesh.io/whitepaper*), which more broadly provides an overview of the opportunity for RightMesh and the RightMesh Token Generating Event. This technical white paper gives more details and assumes in-depth knowledge of complex systems, networking, encryption, and cryptocurrencies. Throughout each section, the general high-level architecture is presented, along with justifications for design decisions. Our progress to date will be presented, and areas where future work is required will be specified in each section. We will conclude with a summary of the ongoing technical roadmap which we believe helps to support the original paper.

The document makes the technical case for a tokenized mobile mesh network where the devices are mostly smartphones, IoT devices, sensors, automobiles, and other devices that have had difficulty connecting traditionally since they may move into (or exist in) parts of the world that are connected to infrastructure poorly--and are likely to remain as such for some time. The token system is designed to incentivize devices and people to join a network, stay within a network, and act as virtual infrastructure. It has been designed so that it can work immediately in today's cryptocurrency environment. That is, the system operates under the assumption that smartphones are still not capable of participating as full crypto nodes. At the same time, decisions to "future proof" the technology have been made with the assumption that this will not always be the case.

MeshID

Given that RightMesh is able to support connecting together multiple Wi-Fi hotspots—each with their own set of IP addresses (both IPv4 and IPv6), as well as connecting via Bluetooth (with MAC addresses), and in the future other networks where there may be even more different addressing schemes—one the first problems we aimed solve is how to identify devices in the network uniquely. While building YO! (one of the company's first offline apps), this was handled in the typical manner where unique IDs are generated from an Internet-connected server. In a mesh where the devices may never touch the Internet, or may not for a long period of time, this makes it impossible to form the identity. It is a chicken and egg problem. With RightMesh, we needed to generate the ID on the device without requiring the Internet to do so. Researching this problem led the company into cryptocurrencies, where this problem had already been addressed and proven to do so in a way where there is almost no chance of ever generating the same ID twice.

Currently, RightMesh generates an Ethereum account using the Ethereum-j library. The account generated is compatible with any popular Ethereum client, such as geth, mist, or myetherwallet. This identity is generated once when the RightMesh library first launches on the device, and remains on the device until the user has removed it manually. This means if RightMesh apps are installed and uninstalled, the same identity will be present unless the user removes it. All RightMesh apps that are running at the same time also share the same identity, similar to how your computer has the same IP address for your web browser, your Slack client, and your games (unless you leave your house and travel to work with your computer and get a different IP address at your new location). With RightMesh, despite all the movement and changing IPs while you move, your RightMesh identity will stay constant. This permanence is how other devices in the mesh will always be able to deliver data to the right device.

The identity on the mesh is the public Ethereum address that gets generated along with the account. The generated account is encrypted using a default password that RightMesh provides (as a proof that the service can do the encryption correctly). It also contains the public and private key associated with the account to ensure it works on the Ethereum network when executing transactions. At launch time, RightMesh aims to have the MeshID functioning such that a user will

be able to set their own password for the encryption so that their account is locked and protected by the password as they would expect with an account generated by any other client.

Ethereum was selected as a platform mainly for its smart contract capabilities which the company uses for incentivizing the mesh (more details in later sections). However, Ethereum could be swapped for any cryptocurrency that operates in a similar manner. There are several drawbacks currently with using Ethereum, such as needing Ether in the account in order to transfer tokens (to pay for the gas), the high cost of small transactions, and the scalability problems that have been seen in some recent ICOs and TGEs. Many of these problems are actively being worked on currently by the Ethereum community. For instance, it will soon be possible to pay for the gas of a transaction with the token rather than Ethereum. There are further enhancements coming that will enable microtransactions to occur more affordably. There have been proposals for side chains, which we initially also proposed, however, after feedback from the initial version of the white papers, the community suggested that the sidechain approach we were intending to use was not the best option.

With the sidechain we intended to release the full transparent chain generated on the side and also allow community partners to run superpeers. Allowing community partners to run superpeers in this model would have allowed a 51% attack, so it would have forced us to run many superpeers ourselves, or enforce that the community could only operate fewer than us. It would also force the community to trust that we would run the superpeers correctly and that we would "approve" new community superpeers in a fair way. We are currently working towards a version that removes the sidechain altogether to address these concerns. This new version will use or be based on µRaiden and Lightning, whereby payment channels are used between the buyers of bandwidth and the superpeers, and sellers of bandwidth and the superpeers. We will likely have to port the python microRaiden client to Java for the RightMesh library. This client will be open sourced so that the community has the benefit of another type of client to use with Raiden and we will also be able to use the same code in RightMesh so that we get the benefit of community improvements on the library as it becomes more widely used.

Initially, we will operate the superpeers so that we can implement the approach more quickly and so that we can at least assume that the superpeers will initially not be acting maliciously. Of

course, the community will have to trust at first that the superpeers that RightMesh will operate will not act maliciously. Then as quickly as possible after this, we will work towards the cases where the superpeers may operate maliciously, and if so, the tokens tied up in the channel are lost as punishment.

We have previously built some proof-of-concept apps which could perform the necessary transactions to support token transfers on a private or semi-public side-chain. These contracts have been open sourced and added to git. All of the portions of our library that touch the cryptocurrency and incentivization side will continue to be open sourced, along with all of the sample apps built by RightMesh, so the community can understand deeply how our solution works and work with us to build something incredible.

RightMesh System Overview

Figure 1 shows the high-level system overview and tries to capture all of the possible layers and roles within our system from the perspective of a data-sharing network where there are some devices that wish to sell their data for RightMesh Tokens (MESH); some that wish to purchase; and some that wish to participate as forwarding (infrastructure) nodes. At the highest layer, we also provide some additional infrastructure that presently cannot be eliminated until it is more feasible to run full Ethereum nodes directly on the mobile devices themselves (and to get around things like firewalls put in place by existing ISPs).

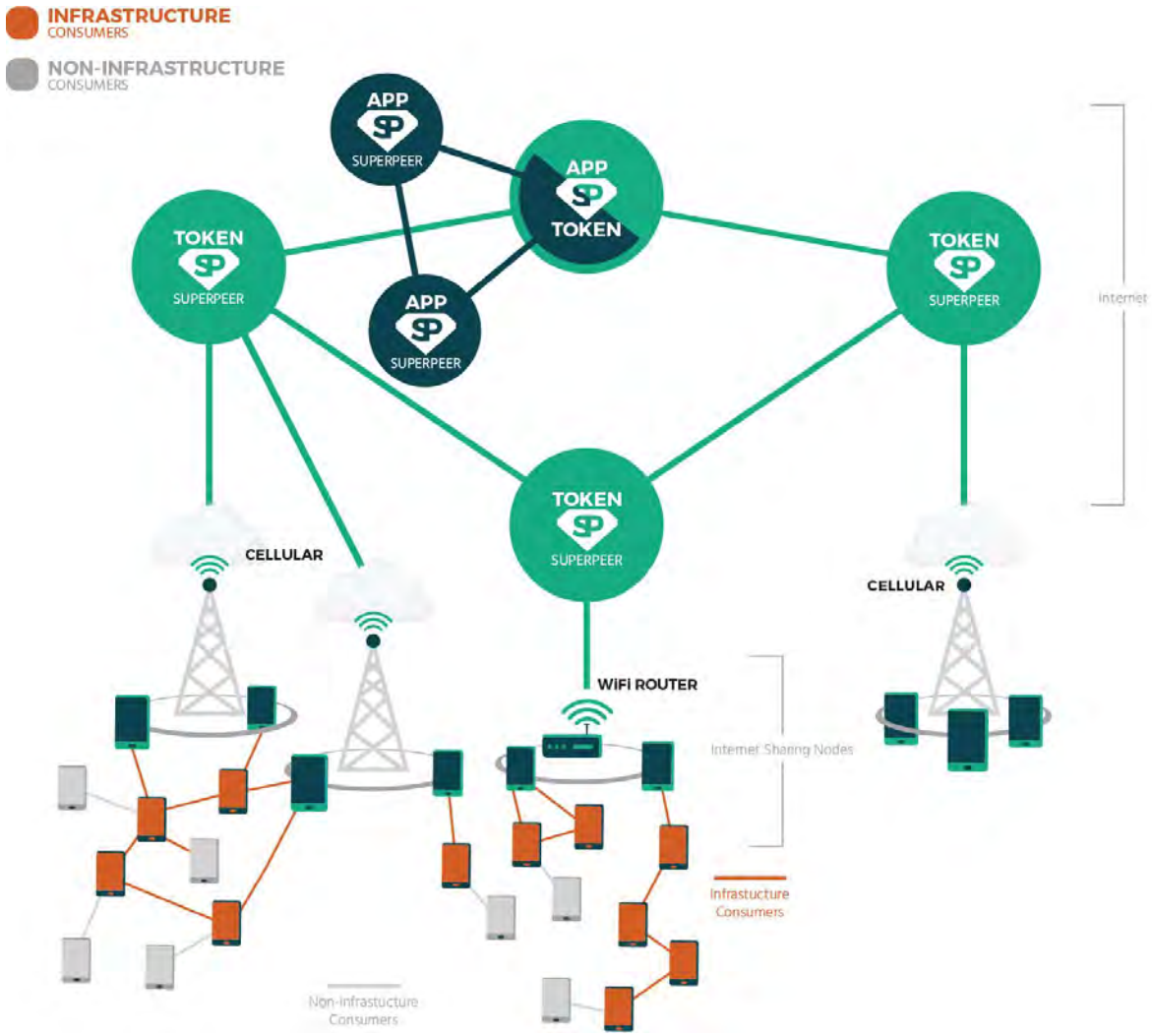

Figure 1: High-level system overview

Roles

In all of the following discussions, the roles and layers are specific only to data traffic. In some cases, devices which are unwilling to forward data can still earn tokens in other ways—such as providing storage, processing, or other resources.

In *Figure 1*, there are a few key pieces and layers to this system. There are token superpeers (labelled as such), which are responsible for the routing of packets between geographically separate meshes, and for relaying Ethereum transactions from devices in the mesh to the real Ethereum network. There are also app superpeers (labelled as such - more details to follow). It is also possible for app superpeers and token superpeers to exist on the same node (although it is not necessary that an app superpeer also be a token superpeer). The phones which are directly connected to cell towers, Wi-Fi hotspots, and other Internet connections are called Internet sharing devices, which may or may not actually share their Internet access. In orange are infrastructure nodes (which may also be consumer nodes). In grey are non-infrastructure consumer nodes.

Token superpeer devices are running full Ethereum nodes, and execute signed transactions on the Ethereum network on behalf of the participating devices. These nodes also act as the main linkage between geographically separate meshes. For instance, on the far left connected to the cellular network, we can consider that mesh to be made of devices in Canada, while the mesh connected to the Wi-Fi router in the middle could be a mesh formed in Bangladesh. Since they are too far away for a single mesh to form, we have to rely on existing infrastructure (ISPs, wires, etc.), and use token superpeers to facilitate this communication. There are two ways in which this can be achieved.

First, the method that is working right now is to simply act as a forwarding node, and directly forward all traffic from one mesh to another. The second method, is to act more as a lookup service. In this case, the token superpeer would be able to determine which other Internet sharing device (in blue) traffic needs to be routed to. After this is determined, the information is sent back to the Internet sharing device on the source side where it will communicate directly with the Internet sharing device on the destination side. This is more complex since it must be able to get around firewalls in order to function on all networks. This second method is left as a future exercise that

will bring our costs down (since we will lower our data costs from operating the token superpeer on AWS or some similar service).

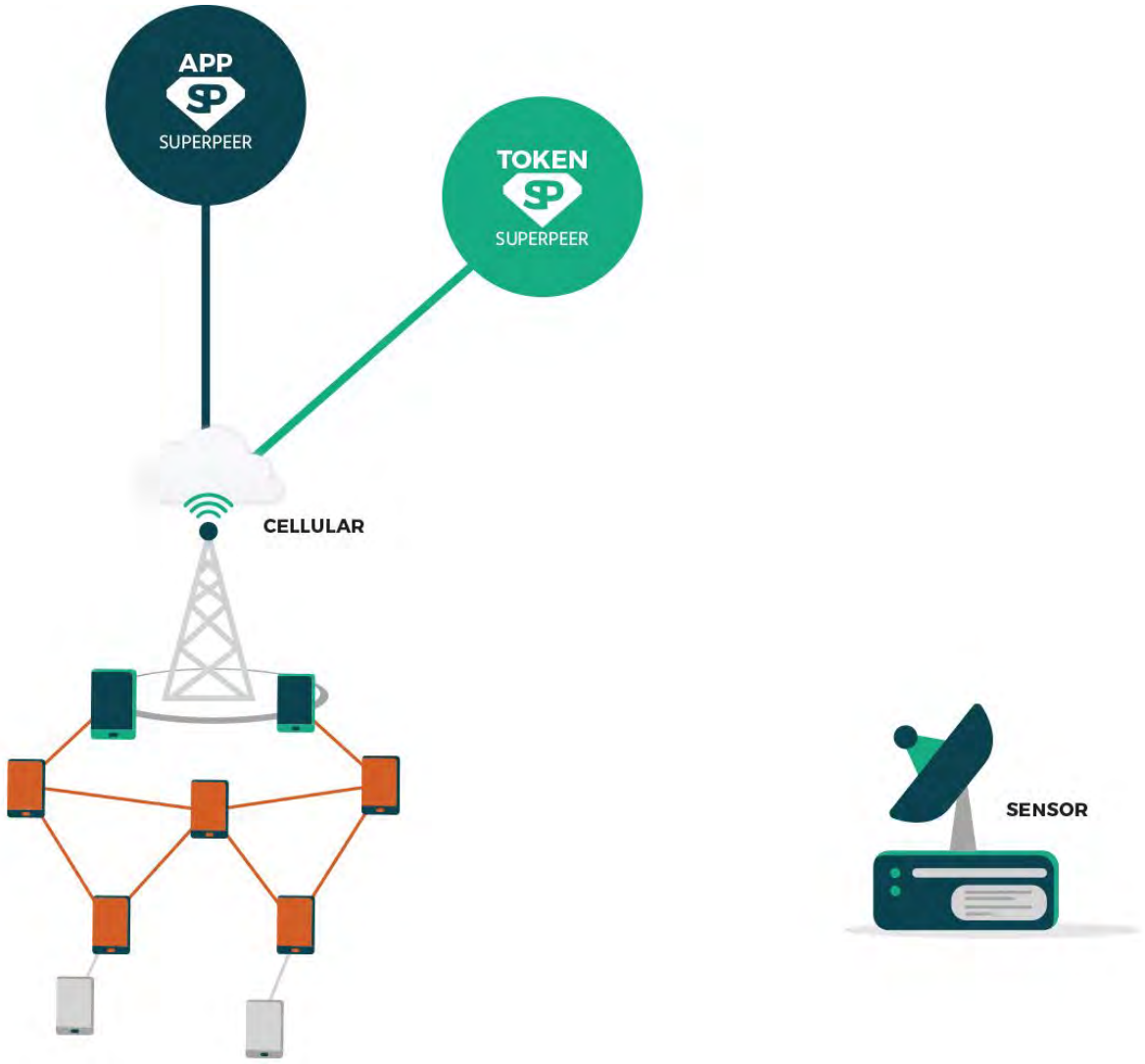

Figure 2: "Data-mule" capability linking geographically separate meshes where some lack Internet connectivity

In addition to token superpeers, it is also possible for app developers to provide their own superpeers. In this scenario, the app superpeer would act as a trusted device (trusted only by the app which is making use of it). For instance, an app which allows people to play geocaching may also deploy a geocaching superpeer. While the app lets people discover geocaching locations, where the location is actually a remote sensor, which relays data into the mesh eventually towards the geocaching superpeer, where the data may be stored and presented on normal traditional websites. This "data-mule" behaviour is demonstrated in *Figure 2*. The app superpeer would be able to provide some further infrastructure needs that a normal mesh app could not provide. For

instance, consider a mesh app where there are remote sensors, but in a location where people occasionally pass (e.g., in the mountains or in northern Canada). An app superpeer would be used as a collection point where the data would be stored for visualization, analysis, etc., and the RightMesh library would handle routing, despite the phones not always being connected to the Internet. In our remote sensor example as a hiker passes by, it would forward the collected data the next time the phone had a connection to the app superpeer through RightMesh automatically. Note that app and token superpeers may or may not exist on the same, physical AWS instance.

Internet sharing devices (SUPPLY) may be connected to the Internet using a cellular network, Wi-Fi, Ethernet (if the sharing device is a router, computer, etc.), and have the ability to share their Internet connection through the RightMesh library. Note: Internet sharing at this time, means that we can use the Internet connection to link to other geographically separate meshes, or to superpeers, but not general purpose Internet access presently. General purpose Internet traffic is on the roadmap, however.

Layers

The app and token superpeers are directly connected to the Internet using more powerful devices than mobile phones (essentially what existing p2p systems require in most cases). We should consider the Internet connection in these cases to be fast and stable, and the devices themselves should be fairly powerful and highly available. The devices should also have the ability to control firewalls and open ports. Currently, these devices would likely be run by RightMesh itself on an interim basis and exist on AWS, Azure, etc. They will also be able to be run eventually by Community Members with systems that match the performance requirements to operate such a node. In the the longer run, these devices will also be located on smartphones with the same characteristics.

The next layer down is the small, blue square devices: the Internet sharing (SUPPLY) devices. These devices are connected directly to the Internet - although they don't have to be all the time. This Internet connectivity decision is left up to the users of these devices. Though the decision may depend on what network they are connected to; for instance, some users may not wish to be Internet sharers on the cellular network, but they will happily do so using their home Wi-Fi network. These Internet sharing devices can have any speed of Internet connection, and the

capabilities of the devices may be much lower. They do not need to run a full Ethereum node. Preferably, these devices are not moving too much since many other devices may depend on them for access. However, if other nodes do depend on them for access, a moving device is still better than nothing. Currently the RightMesh library selects the routing path to Internet devices based on hopcount and cost, however eventually the RightMesh library will be able to select which path to take eventually based on how mobile it is, while prioritizing those that will result in the best performance for the lowest cost for everyone.

The next layer is the orange squares. These squares indicate devices willing to provide infrastructure. These nodes may also be consumers (DEMAND). The infrastructure nodes do not have an Internet connection themselves, so if they wish to use Internet data that isn't being forwarded on behalf of another, they will also need to pay an Internet seller in RightMesh tokens.

There is also a special type of infrastructure device, which is the orange device in *Figure 2* that showed interfacing with remote sensors. This device is a mobile piece of infrastructure where it is providing forwarding as a result of its mobility. These devices may have an opportunity to earn extra incentive due to the effort of physically moving to provide the data delivery. This could be used to offset costs to travel to the remote location such as gas, or just time and effort for a hiker.

The final layer is the light grey devices. These devices are those which do not wish to provide infrastructure, and they solely want to consume (DEMAND) data on the network.

Mobile / Android Architecture

On Android, the RightMesh library functions as a service. Multiple apps can make use of the service at the same time, provided they are all listening on different mesh ports. The RightMesh Developer's Portal (shown in *Figure 3*) allows developers to register keys which are associated with each mesh app they build. When registering the developer key, they also pick which mesh port(s) the app will be listening on and verify that no other apps are using the same key. The developer who created the key may share access with other developers, so teams can collaborate on the same app without sharing login information. If one of the developers leaves the project, their access to generate valid keys may be removed. Farther down the technical roadmap, the developer

portal will also be used to help developers debug mesh connectivity through visualisation, file bug reports, deploy updates through the mesh, and as a mesh analytics platform.

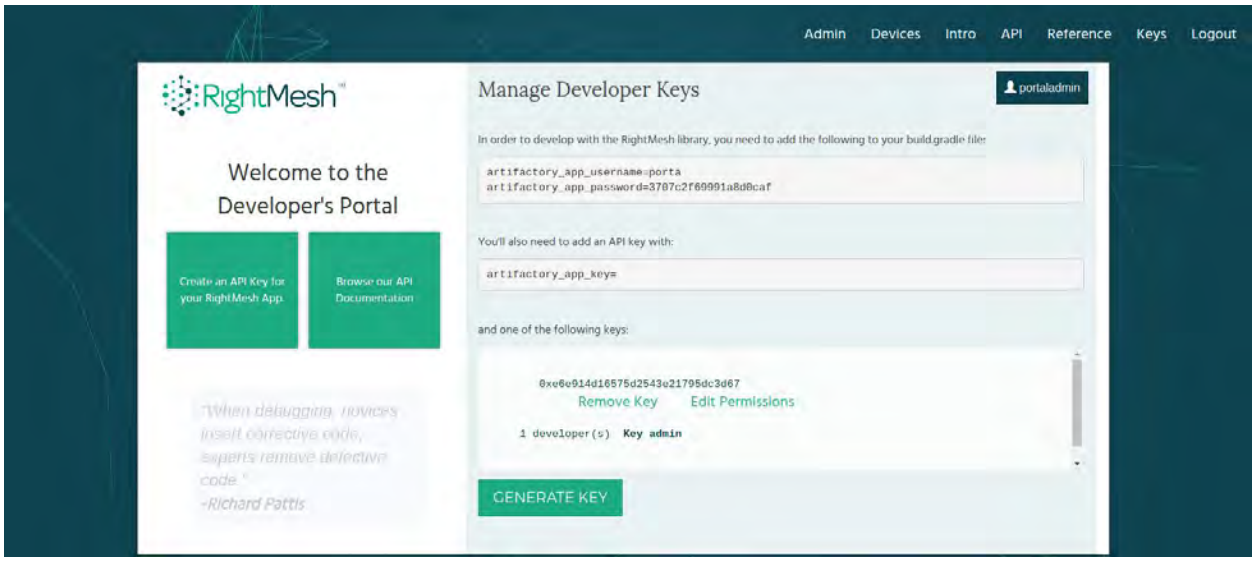

Figure 3: RightMesh Developer's Portal

When the app connects to the MeshService, the service parses a cryptographically-signed license key that is generated at compile time and packaged with the app by our RightMesh Gradle plugin. (An example Gradle file for compiling with RightMesh can be found on our "HelloMesh" project on GitHub. The signed license key file requires a valid developer account with access to the key at compile time. Inside this license is the developer key, and mesh ports that the app is allowed to use. Again, similar to the MeshID problem, we tried to design this system so that apps may form meshes without requiring Internet access to validate the library (although we make the assumption that Internet access is available at compile time for the developers).

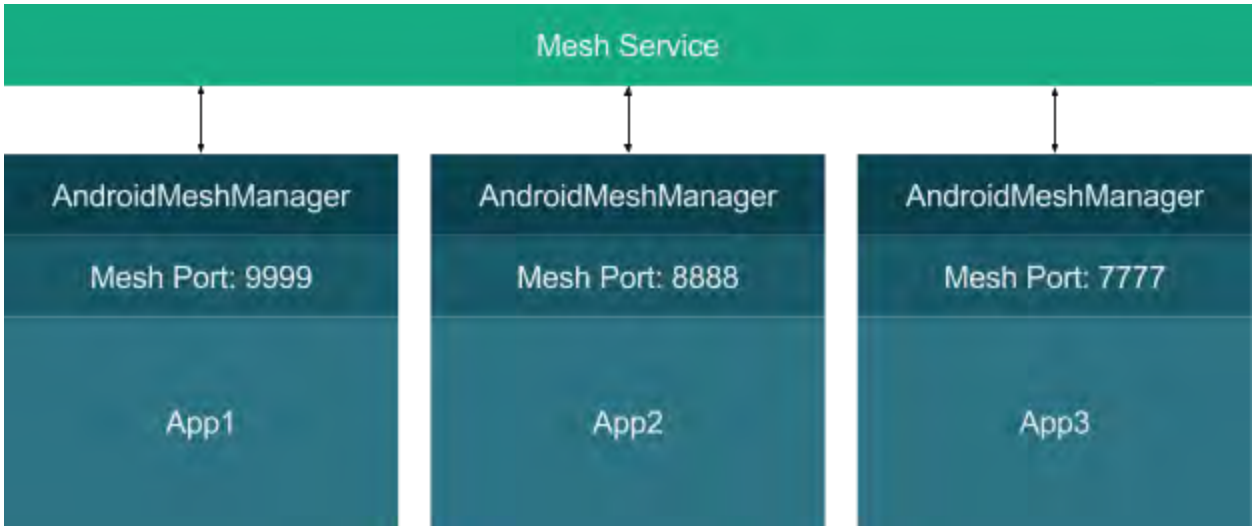

Figure 4: Multiple Apps running on the same device, using a common service

Figure 4 shows the RightMesh Service running on a single phone, with three different apps running at the same time. They are all running on different mesh ports. Each app also comes with a portion of our library we call the AndroidMeshManager, which performs all of the logic to connect to the service, and to fire events such as Peers joining or leaving the network, data arriving, or the result of encryption keys being exchanged.

In Figure 5, a mesh with three devices is shown. Notice that device one has three apps running. Apps are only notified of peers which are running the same app (the developer does not get access to all of the devices on the mesh, even though other devices may act as forwarding nodes in order to reach other devices). For instance, App #1, listening on mesh port 9999 will only "see" peers 1 and 3. App #2, will only "see" peers 1 and 2. App #3 will "see peers 1 and 3 as well. Data being sent from peer 1 to peer 3 will be sent through peer 2, even though it isn't running either of the apps that peer 1 and 3 are using.

An example app which can do basic user discover, send data, and receive data is available on our "HelloMesh" GitHub project.

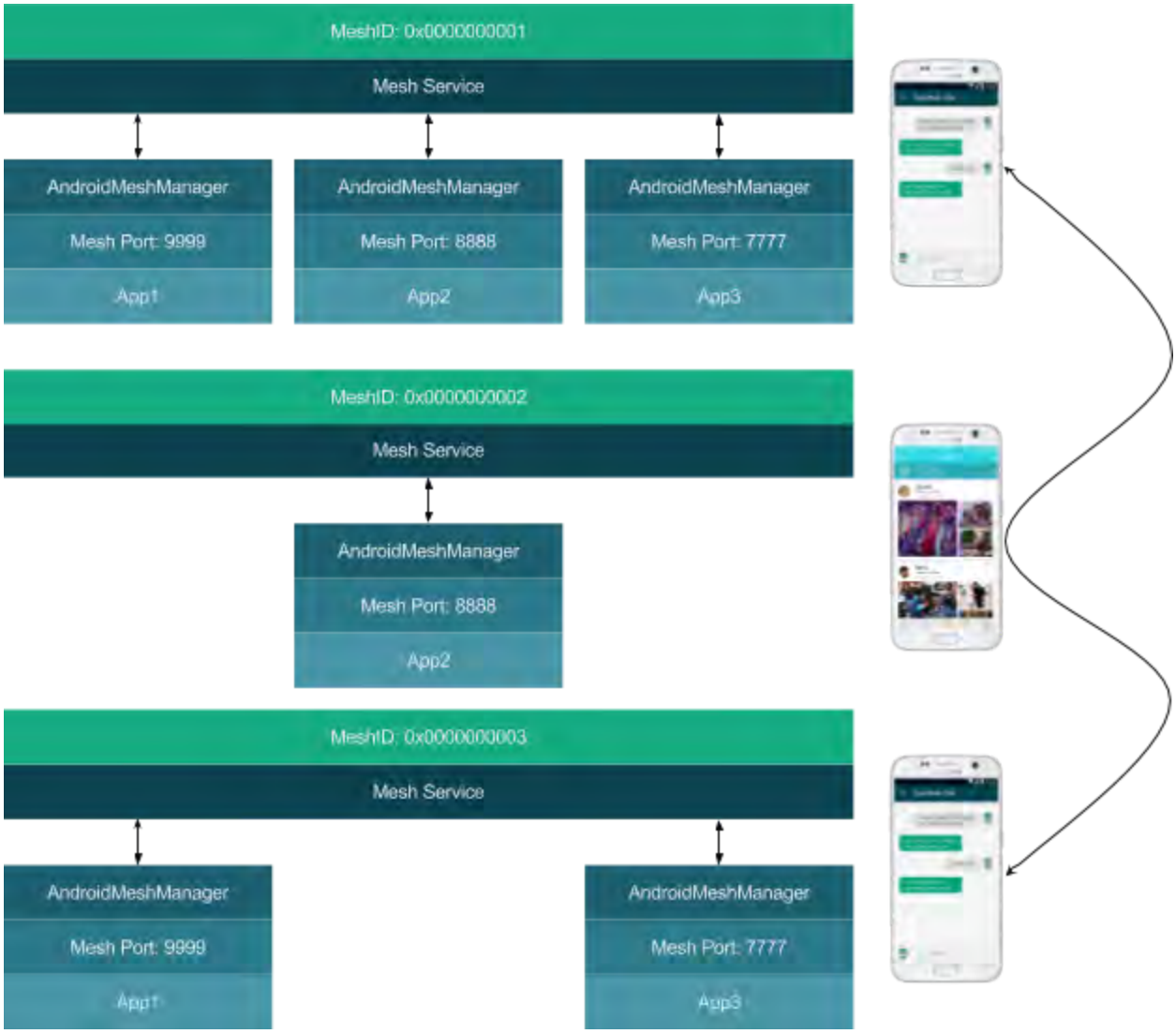

Figure 5: Multiple devices communicate through the service, where each phone may have different apps

Java Architecture

The Java architecture is less developed at this moment, and does not provide a service architecture. As such, only a single mesh app can run on the Java device currently (although turning it into a service is in the works and is on the roadmap). The Java architecture is mostly used to implement the superpeers right now, although it could also be used on IoT devices, routers, and other less mobile equipment.

We have also done proof-of-concepts at hackathons where the Java version acted as an Internet sharing node. It is also quite easy for it to be an infrastructure-less node. Performing tasks—such as

linking together Wi-Fi, Wi-Fi Direct, and Bluetooth networks—is somewhat platform dependent and is still in progress at present. The roadmap aims to support Linux and Windows in the near term.

As of version 0.4.0 of RightMesh (December 2017), we now support relaying data from one geographic mesh to another via the superpeer. The source of the superpeer is available on Github. We have also recently added a HelloJavaMesh Java app which can communicate from a computer to phones running the HelloMesh app on Android. This can work either through the superpeer, or if the computer connects to a phone in Hotspot mode. We are still working on integrating the Raiden network, but expect to have this completed around the same time as token generating event.

We also support visualizing the connectivity of the global mesh with the token-superpeer currently (Note: in the link, it shows up to 100 Internet sharing nodes connected to an early version of our superpeer. There is also a video that shows phones coming online in our visualization tool). This will help as the network grows to debug connectivity issues.

RightMesh Networking Stack

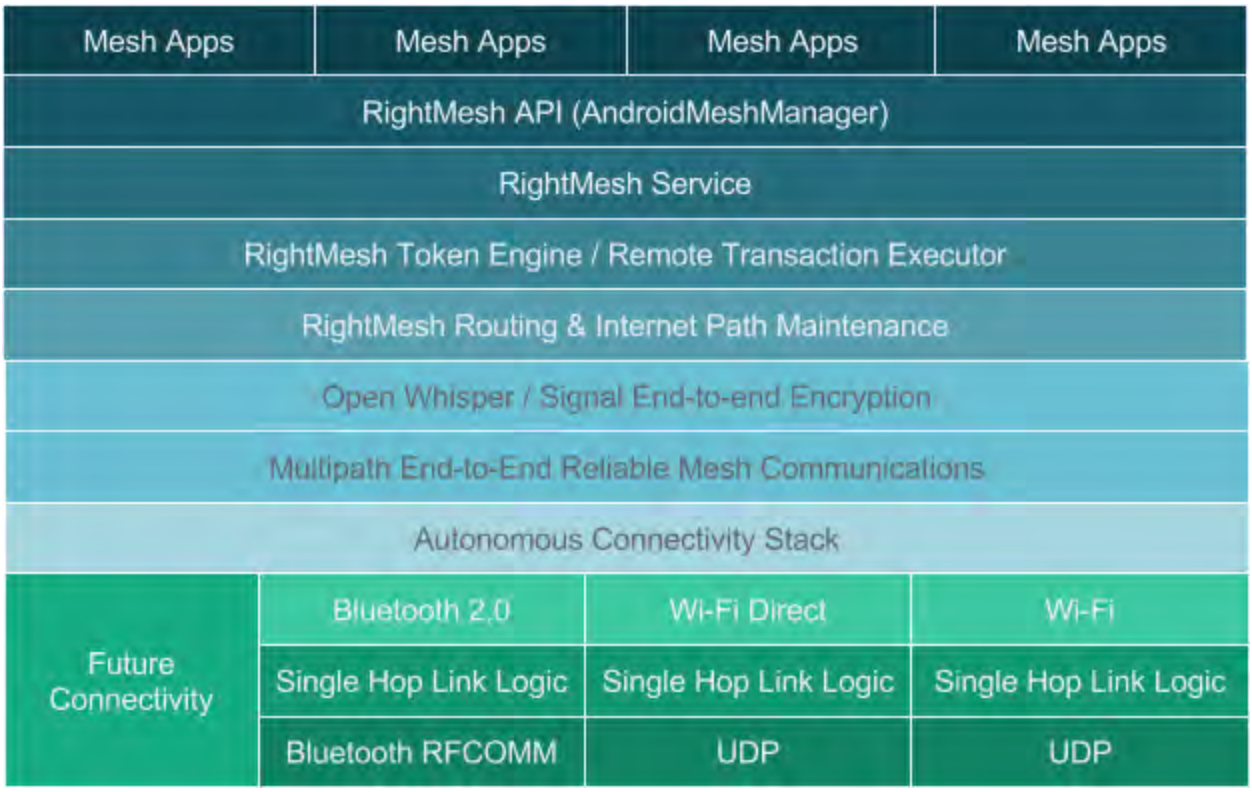

Figure 6: High-level system overview

Layered Architecture Similar to the Internet

The RightMesh networking stack shown in *Figure 6,* builds off of the success of existing Internet protocols. The design is a layered design, so improvements can be made iteratively on each of the layers without having to do a full redesign of the entire system. Everywhere in RightMesh, next-hop connectivity is formed without user intervention. This is a core guiding principle of RightMesh: the user should not have to approve every single device connection (as one has to do with Bluetooth pairing, for example). The only way a widespread mobile mesh network will scale from a user experience point of view is if the connectivity can be made without user authorization of every connection. If the user is required to approve every single connection to every other device in the network (how some similar libraries work), it would be difficult to get meshes that scale beyond a few devices locally - and most people will be inclined to only form meshes with people they directly know.

One-hop Connectivity

Bluetooth 2.0 was used as a starting point because it was possible to make a connection with other peers without requiring pairing. With our Wi-Fi Direct and Wi-Fi implementations, the same is also possible. RightMesh makes connections programmatically using three different types of next-hop links. Within our design, RightMesh can add support for higher versions of Bluetooth as we figure out ways to make the connectivity occur without user intervention. We may be able to do this by making use of some of the other connectivity pathways we already have as a way to signal information between devices to set up the faster connections.

Everything below the autonomous connectivity stack can be thought of as similar to a layer two protocol in a normal networking stack (think of MAC addresses being mapped to IP addresses). Instead of that, our protocol maps IPv4, IPv6, MAC, and other addresses to a MeshID. This layer is responsible for exchanging local peer information. We make use of a clustered routing where hierarchy is imposed on the devices based on the internal roles the devices are assigned. This means that not every device needs to know the connectivity to every other device, allowing the RightMesh network to scale significantly higher than competing approaches. This same cluster based architecture lends itself well to enabling caching in the network which is also on our roadmap. For instance, the same nodes that are cluster heads would be ideal candidates to also be a cache of common apps, ads, and multimedia content.

Autonomous Connectivity

The autonomous connectivity layer does the job of assigning internal roles to devices in our network. This decides whether a device will be in hotspot mode or not, which devices to connect via Bluetooth, whether to use Wi-Fi, Wi-Fi Direct, both, or all three. The developer does not need to worry about how the autonomous connectivity works. For the developer as far as they know, there is a list of MeshIDs that are either connected, or not, running the same app.

The RightMesh approach to autonomous connectivity is something that will constantly get better, but it was kept simple as a sort of MVP state. In fully-automated mode, the user will only need to select whether they wish to participate as a forwarding node, and whether they wish to share their Internet, or not. If they wish to sell their Internet, they will also need to set the price they wish to

sell it for (in Mesh Tokens). Conversely, if the user wishes to consume Internet, they must set the price they are willing to purchase Internet for (in Mesh tokens). If they do not wish to pay any mesh tokens and there are no free paths, the apps will still function, but they will only be able to use a localized version of the mesh.

It is also possible, in advanced mode, for users to control exactly which internal role the device should use. This capability is packaged automatically with every app that uses the RightMesh library (see *End-Users* section).

Multipath, End-to-End, Reliable Mesh Communications

Combined with the routing layer and the one-hop connectivity layer, multipath end-to-end reliable communication forms a really unique and valuable part of the RightMesh solution. This is what distinguishes RightMesh from most of its competition. There are a few ways communication typically occurs in mobile meshes:

1) Broadcast to everyone (either using UDP + existing broadcast / multicast approaches, or TCP on a link by link basis - no end-to-end capability
2) Single-path end-to-end TCP (with many limitations as outlined below)
3) Best-effort UDP communications with no reliability

Many existing mesh approaches simply make use of existing TCP to provide end-to-end connectivity. However, TCP is not well suited to a mobile mesh network for several reasons:

1. Unless rooted, mobile phones do not let the user customize the version of TCP being used.
2. Mobile mesh networks are made of devices that frequently connect and disconnect. When this occurs, a TCP connection would require end-to-end reestablishment of a connection, with something like a 3-way handshake.
3. With IPv4 It is often impossible to make an end-to-end TCP connection because the IP address in one hotspot means nothing to the neighbours three hotspots away. As such, there is no mechanism to exchange this information between hotspots and have the routing automatically work. In this case, it may be possible to have TCP on each single-hop link, but there are still the same drawbacks from some of the other bullet points to be aware of. In the case of performing TCP on each link, there is still no mechanism to prevent an entire end-to-end path from being saturated with traffic (the congestion control will

only work on a link-by-link basis, leaving the queues at the intermediary nodes to potentially overload).

4. TCP often gets interference and congestion mixed up. It may start a backoff unnecessarily thinking that congestion is occurring when it is temporarily poor network conditions due to external factors (e.g., a vehicle driving through the path of the signal, a microwave being turned on, a tree being between two people who are moving, etc.).
5. The TCP on most non-rooted phones cannot handle multiple paths. This means even if your device has a Bluetooth, Wi-Fi, and Wi-Fi Direct connection available, it is only able to use a single one of them for one data stream. This also applies to Internet connections out of the mesh as well. There are existing versions of TCP that can do this, notably mTCP; however, it is almost never included in commercial phones.

RightMesh borrows concepts from delay-tolerant network protocols such as LTP and has combined them with multipath-TCP to create a delay-tolerant protocol when the network becomes fragmented, but high performance in the cases where it is well connected. RightMesh also uses concepts from Software Defined Networks, Overlay networks and self-* networks.

Open Whisper / Signal End-to-end Encryption

Rightmesh supports end-to-end encryption using the Open Whisper/Signal library (whispersystems.org). This library has been modified so that it no longer involves the server portion, since that would require Internet access. RightMesh offer two levels of security: one where the key is directly exchanged in one hop (this is the more secure option). The second, where the key exchange occurs through the mesh over several hops, is less secure because it may be subject to a man-in-the-middle attack.

RightMesh does not store any keys in any server, so any key exchange that occurs securely means only the recipients can decrypt the data. For the company, there is no way RightMesh can be compelled to give up the keys because none are stored. We are working on ways to improve this process (e.g., like sending the key split up across multiple paths so that any attacker would need to compromise many devices at the same time). The company is also working on ways to improve the user-friendliness of a secure key exchange such as with a 2-D barcode or NFC. Compared to other

mesh platforms which broadcast to every device, RightMesh only forwards directly on a routing path. As as result, fewer devices have data flowing through them, making it much harder to attack.

RightMesh Routing & Internet Path Maintenance

As part of our peer discovery protocol, information regarding possible paths to the Internet are provided to devices in a local mesh. Currently, the library supports a single path to the Internet, but because of our multipath transmission support, it will eventually be possible to also support multiple paths to and from the Internet simultaneously.

RightMesh Token Engine / Remote Transaction Executor

Since devices on RightMesh have an identification based on an Ethereum account (including the private and public key), using the Ethereum-j library, it is possible to generate signed transactions from the account and pass them through RightMesh to the token superpeer. The token superpeer runs a full Ethereum node. When the superpeer recognizes that a remotely signed transaction is being passed to it, the superpeer executes the transaction on its local geth instance and waits for the confirmation. It then passes the result back to the device in the network which executed the transaction. This means that any device within a RightMesh network is able to send Ether to another device, query its own Ether balance and execute contracts on the Ethereum network from potentially many hops away from the Internet. At best, as far as we know, thin clients exist on mobile phones which can perform similar capabilities when the device is connected to the Internet directly (similar to how the blue Internet sharing devices are). However, nothing exists that allows for cryptocurrency transactions from outside of the range of traditional Internet access.

Furthermore, the token superpeer is able to execute token transactions on behalf of clients. More information about this is found in the section: *Token Integration & Token Superpeer Architecture.*

Token Integration & Token Superpeer Architecture

Managing limited resources is a known hurdle for RightMesh, whether in regards to battery life, network capacity, or memory and storage limits. Because of these constraints, it is currently not feasible to run a full crypto node on the mobile phone itself. To allow for Ethereum transactions to occur, the nodes would sign a transaction locally and relay it in towards full Ethereum nodes (i.e., Superpeers), running at the edges of the mesh. Some of these nodes would be run by RightMesh and some by community partners. This model may evolve into an industry on top of our ecosystem since these nodes can earn tokens by performing a mining-like function to keep the network operating. At launch, these Superpeers would be run on computers or on existing cloud providers spread geographically around the world. In the future, as mobile devices improve, we believe it will be possible to eliminate this centralized aspect, as much of this functionality may move directly into the Internet Sharing Devices themselves. At present, the biggest limitation is hardware, as it is not feasible for phones to act as full Ethereum nodes. The Superpeer functions as a proxy, a hole-punching aid, a translator between the IP-based Internet and our mesh protocols, and an Ethereum full-node.

All the code that makes use of Ethereum technology will be open sourced, along with any contracts, so that the community can verify and improve upon the stability of the platform and to encourage the users to trust that things are operating correctly. RightMesh will also be open sourcing some of its own created apps to demonstrate possible implementations of the Mesh SDK: https://github.com/RightMesh. The company will explore open sourcing its core mesh technology once the network becomes hardened and the network established. From a developer's perspective, they simply need to define the transaction they wish to make within their RightMesh application, and issue a call to our library. RightMesh will perform the transaction signing and optimize how to route the packet out of the mesh and into the Ethereum network. The full Ethereum node at the edge of the mesh is also running a Java version of our library. It uses the local remote procedure call (RPC) mechanisms to relay the transaction to a locally running Geth client. The Geth client will verify the transaction and execute it. The result of the verification and execution will then be relayed back to the mesh node which originated the transaction.

Using a similar approach, RightMesh could adapt to support other cryptocurrencies as well, such as Bitcoin.

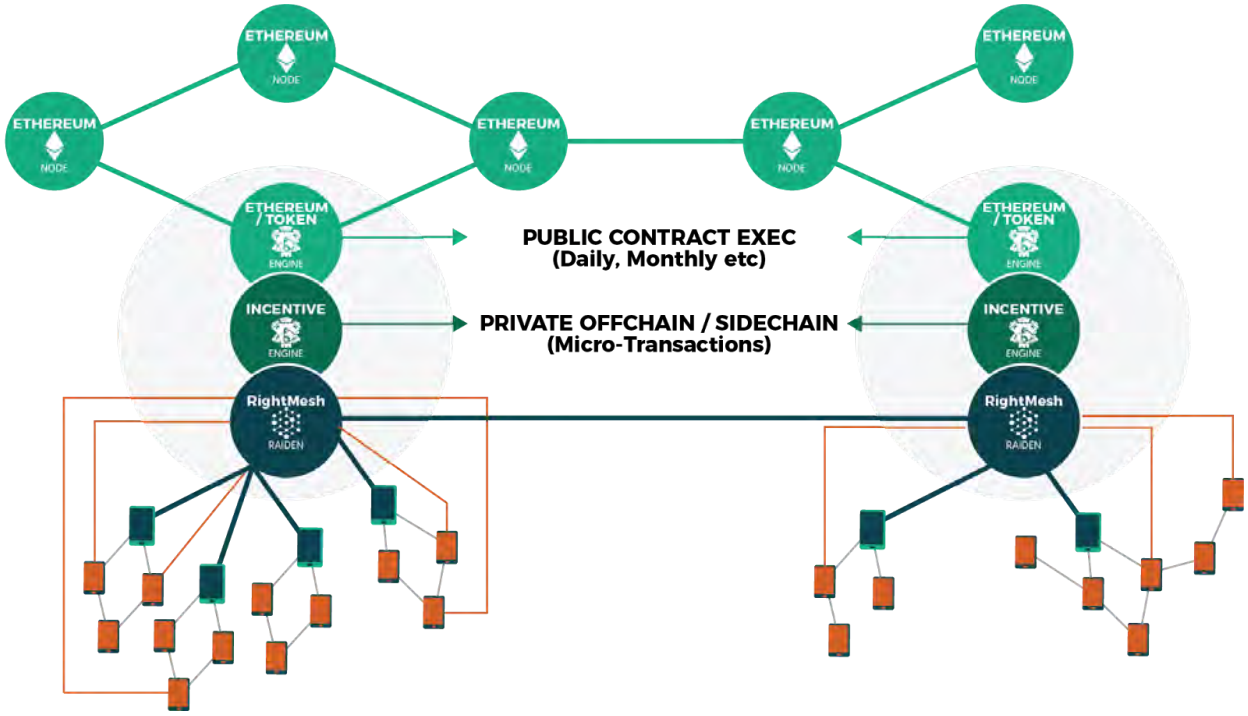

Figure 7: Multiple "token superpeer" architecture to support incentivization of Internet sharing devices

Figure 7 shows a RightMesh network with two participating token superpeers. In the full, high-level diagram at the start of the document, there also may be app superpeers, but these are left out of this figure for simplicity. In this figure, the left superpeer has two geographically separate meshes attached to it while the right superpeer has two different geographically separate meshes attached to it. The superpeers may be deployed all around the world, similar to how AWS instances are deployed in different places based on geography and demand. The left superpeer may be deployed in Canada, and have a mesh in Vancouver and a mesh in Toronto, while the right superpeer may be deployed in Bangladesh and have a mesh in Dhaka and a mesh in Khulna. The green line represents a fast internet connection between the two superpeers. This is important because in the initial implementation, the data will be relayed directly between the superpeers to reach the farther meshes. In addition, the superpeers are running full Ethereum nodes. On the Ethereum network, the two token superpeers may also be peers of each other. In addition, there may be many public peers on the Ethereum network.

Device Roles for Tokenization

Recall from the system overview, that some portion of the devices provide SUPPLY to the network: these are the Internet sharing devices. In the above figure, these are the devices connected directly to the token superpeers. Devices which aid in delivering the SUPPLY to the consumers (DEMAND) are called infrastructure nodes. The consumers may be scattered throughout the mesh, and at times they may be both consumers and infrastructure nodes. At launch, incentivization will be possible when a device wants to send traffic from one separate mesh to another through the internet sharing devices via the superpeers. When we enable general purpose Internet traffic, any device which relays any Internet traffic will be able to be incentivized as well.

Buying and Selling Resources, Interaction with RightMesh

A device user requesting the resources, whether a client device or service provider, sets how much they are willing to pay to consume a certain number of bytes of traffic. This is the DEMAND side of the marketplace. For instance, people around the world are fairly familiar with the concept of a data consumption rate whereby a user may pay for their data consumption to their ISP or mobile carrier, typically priced per MB of data. This price is set via free market with a minimum floor, where the floor includes the base commission for RightMesh services and for Superpeers who execute the RightMesh token transaction.

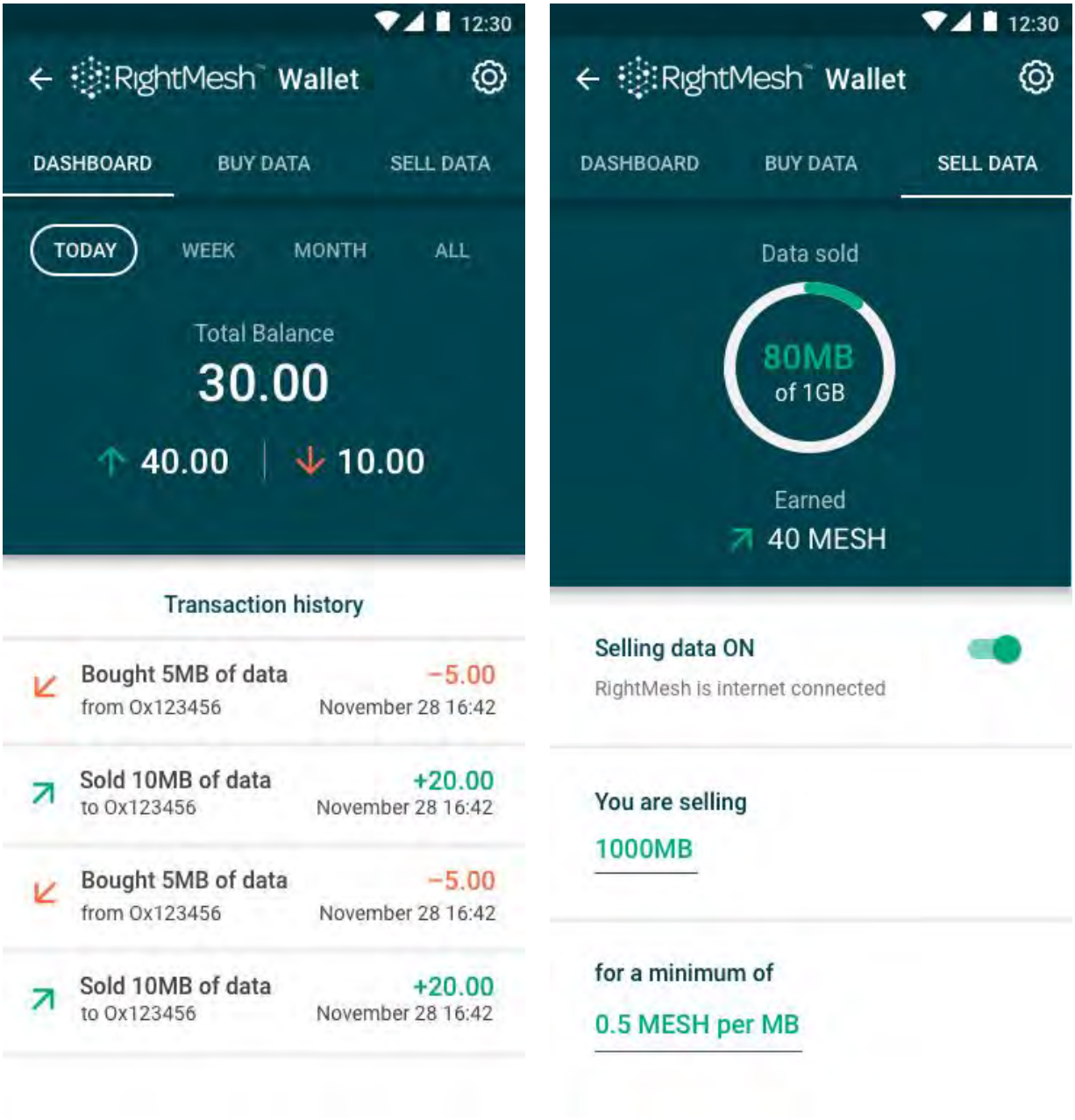

Figure 8: Example of Resource Accounting Built into RightMesh

Similarly, the device users—which wish to participate as an Internet sharing device, as a caching device, or mobile delivery device—are able to set their minimum selling price for their resources. To start, it is likely that only buying and selling data within RightMesh will be supported; however, the roadmap also includes possibility to incentivize app and ad distribution; storage on the phone (similar to, or potentially in partnership with Filecoin or like services); usage of the processing

resources (similar to, or potentially in partnership with Golem or like services); and enabling token transfers between devices on the mesh. This is why the word *resources* is used instead of *data*.

Incentivised Routing in RightMesh

The RightMesh library would interact with the incentivization model such that it would always be searching for paths to the Internet to minimize cost and maximize performance. In a local mesh, RightMesh selects routes based on the least hop count path. This was chosen because it was very simple to implement and performs well enough for a proof-of-concept network. When tokens are involved for the Internet transfers, there are three strategies that are user configurable. First, is local only (i.e., don't spend any tokens). Second is shortest path (i.e., if there's both a local path and an Internet path to the same device, take the shortest one, even if it costs tokens). Third is cheapest path (i.e., in the same case, take the free path, even if it is longer). As our routing protocols become more sophisticated, shortest path would be replaced by "best performance" which would be a derived metric using a formula as in Eq 1.

$$Estimated\ Performance\ Metric\ = \sum_{i=0}^{i<|n} w_i m_i$$

Where:
n is the number of metrics we are interested in (delay, throughput, jitter, link load, etc, etc.)
m_i is the i^{th} normalized metric value
w_i is the weight corresponding to the importance of the i^{th} metric
$w_1 + w_2 + \ldots + w_i + \ldots + w_n = 1$

In other words, convert each metric value into a score between zero and one, and then assign weights of importance for each metric, such that the weights sum to one. The end result is that you will end up with a score between zero and one that represents how good the route might be.

This could further be extended to include traffic classes so that real-time traffic would have different weights in this function compared with streaming traffic, file transfers, emails, and other best-effort traffic. [1]

Mesh Token Channels

In the previous version of our technical white paper, we described a sidechain system where control was still centralized in some ways. The main motivation for this previous design was to avoid the high cost of settling transactions directly on the main public blockchain. Since then, we have received community feedback and the Raiden network has launched on the public Ethereum network, which aims to solve this exact problem. We have since redesigned the tokenization model of our protocol using this type of Lightning / Raiden approach whereby micropayment channels can be established between devices participating in the network. As such, it is our intent to establish payment channels between each RightMesh device and the superpeer which would facilitate transactions between the buyers and sellers. Note: we likely can't directly use the Raiden library directly because it is implemented in python and our library is written in Java; however, we will heavily be consulting the code base and making use of the smart contracts which have been developed specifically for a Lightning-like network on Ethereum.

Why Involve the Superpeer?

Consider a network where we don't involve the superpeer, and just set up payment channels between the buyer and seller directly. This would be problematic because it would require a clean teardown of the channel every time the connection point to the internet changed (i.e., every time a buyer switches sellers). Since the underlying routing protocol may opt to do this as a particular buyer runs out of available bandwidth, or turns its connection off, or simply leaves, there is no guarantee this could be accomplished. This would leave the buyer with all of its funds potentially locked into the channel for some time. It would also make it difficult to provision the appropriate amount of funds to use multiple sellers at a given point in time. In other words using a superpeer to

[1] For extensive reading on these subjects consult with RightMesh's Chief Networking Scientist, Dr. Jason Ernst, who wrote his PhD thesis on this topic: "Simulation Models and Framework to Support Connection point selection in HWNs to improve QoS/QoE, reduce cost and increase profits". These types of approaches are also common in the literature with techniques such as TOPSIS and MCDAS being just a couple examples.

facilitate means the underlying data routing in the network may change significantly without any impact on the payment channels. As long as the buyer device is still connected via the same superpeer(s), the payment channel would remain unchanged. In addition, the superpeer provides a vital service to the Internet Sharing devices - which is to enable them to bypass NAT and firewall connections - so this service is worth some service fee as a result.

To get around these limitations, we propose that the superpeer is used to facilitate the sales between the buyers and sellers. This means each device participating in the RightMesh network only needs to create one payment channel with the superpeer directly. The superpeer would be required to hold many tokens in order to scale because for every connection that wishes to buy or sell data, some amount of tokens would need to be locked up for the lifetime of the channel. However, this isn't really a limitation - it is actually quite desirable. It gives people who hold large amount of tokens good reason to participate in the network in a meaningful way. As part of operating a superpeer and being willing to temporarily tie up some proportion of tokens so that other people can buy and sell data means that fees can be charged by the superpeer. When the superpeers are operated by the community these fee prices can be market driven. Some people may wish to offer the service for free, others may wish to charge some money. End user phones will select superpeers based on cost and performance. Superpeers with free fees will likely end up with all of their available funds being locked in channels and will not be able to service any further buyers, so there will be scarcity in terms of accessing the cheapest superpeers.

Granularity of Payments

In our previous proposal, despite using the sidechain, users were still required to reserve chunks of data. The size of the chunks would have been set by us, after some experimentation. The reason for this was because we didn't want the complexity of contract lookups on every packet. Now that the payment model is significantly simplified by using channels, it is possible that we can append payment information to every data packet.

In the uplink direction, for example, the buyer would send its first data packet and with it attach its half-signed commitment transaction to the superpeer, through the seller device (the Internet Sharing Device). The Internet Sharing device can verify that the payment is indeed being sent and forwards the packet to the superpeer. The superpeer prepares an ACK for the data packet and

inside the ACK attaches its half-signed commitment transaction to the buyer, as well as a half-signed commitment transaction from the superpeer to the seller.

In the downlink direction, data arrives at the superpeer from a different geographic mesh. The superpeer determines which Internet sharing device to forward it towards and issues a half-signed commitment transaction charging the buyer as the data is being sent into the Internet link. If the Internet sharing device detects packets which aren't being paid for, the superpeer isn't behaving correctly, and the Internet sharing device may choose to join a different superpeer. When the buyer receives the data, it responds with an ACK to the superpeer containing the half-signed commitment transaction agreeing to the payment.

Ideally, these channels stay open indefinitely because closing the channels causing fees to be charged to save the updated final balance on the public blockchain, but there are cases when someone may wish to close their channel. There are a few limitations to adding the payment information on every packet. First, it significantly increases the packet overhead on Internet packets. Whereas previously our headers may have accounted for somewhere in the range of 128 bytes or so, attaching the ethereum transaction to every packet header may be too much. Secondly, the signature operation may take too long. If a signature has to be produced for every packet, it could significantly slow down the processing speed of packets. This could be mitigated in a few ways. First, the maximum packet size could be increased, which would lower the overhead relative to the actual data. However, since we are using UDP between the Internet Sharing Device and the Superpeer, this would result in a lowered likelihood that the packets would actually arrive successfully (as the packets get larger, they are more likely to encounter links which will split them up, and then it is required that they all arrive without being lost and in order on their own. To mitigate this problem, we could switch the Internet link only to a TCP link, since we can always assume that the Internet link itself will have a public IP address. A second way to address this problem is to temporarily queue up outgoing sends of data. Lastly, we may not wish to issue a signature on every packet, but every so many bytes of real data instead, or we could tie it to the sliding window mechanisms of our protocols. Similar to congestion control algorithms eventually send more and more data until they hit congestion, it may be possible to send along the signature at the start of every "transmission window" so that it is directly tied to network performance. Likely

initially we will tie the accounting to every packet, and then adjust this as we experiment with performance as the network scales.

Potential Issues Closing Out Channels

Keep in mind that channels remain open and active when devices disconnect and reconnect to the mesh network. The channels are formed directly between the phones and the superpeers on the mesh. This topology of payment channels was chosen rather than directly making a channel between buyers and sellers because it would require the buyer to lock too many tokens in a channel with a buyer who may go offline at any time. The superpeers which are on a stable and reliable network should be reachable far more often so all phones should use the superpeer to relay the payments between the two parties.

There are some interesting issues, however, in the cases where devices disconnect, or attempt to act maliciously. The tricky part will be distinguishing malice from connectivity issues. Assume the case where a buyer is purchasing data from a seller. After some time there will have been many pairs of signed transactions exchanged between the buyer and the superpeer (which is taking payment from the buyer on behalf of the seller). Eventually the superpeer may wish to close the channel to recoup the tokens in the channel. The best case scenario is that some cleanup channel communication occurs and both agree and broadcast the most recent set of signed transactions. In this case, the superpeer gets the tokens almost immediately from the channel closing. However, assume that the buyer leaves the network suddenly and the cleanup is not able to be done cleanly. Since the superpeer would still have the most recent signed transaction, there is no danger of the superpeer losing tokens, however, it would be locked up until the channel times out (or until the device comes back online and transmits its signed transaction).

Another similar case is on the seller side. In this case, the seller has made many sales and has agreed on signed transactions with the superpeer (which has fronted the tokens from potentially many buyers). Eventually the seller may wish to close the channel to cash out her tokens (note: this is only really required if the seller wants to spend their tokens outside of the RightMesh network directly - if she wishes to later purchase Internet access somewhere else with the tokens she has earned selling or forwarding data, she simply has to leave them in the payment channel with the superpeer). When she wishes to close the channel, she must wait for the superpeer to transmit the

signed transactions so that the tokens are available immediately. If this doesn't happen, she will have to wait until the timeout on the channel to receive the packets. There is a case where, if the superpeer knows that the seller has disconnected and may not connect again to the Internet before the channel timeout, he may wish to transmit an outdated signature in the hopes that the seller won't come back online in time to produce the more recent signature. Further, if the superpeer is constantly paying in the direction of the seller, and the channel is left with nothing for the superpeer and everything for the seller, there is no incentive for the superpeer not to lie. At worst, the seller comes back online and shows the superpeer is a liar, but there was no remaining money in the channel left to be lost anyway. This can be addressed by ensuring a minimum stake in the channel by the superpeer so there is always something to lose by lying. In the meantime, we will operate on the assumption that the superpeer will not lie so that we can avoid some of this complication.

Tracking Data Forwarded and Payouts to Intermediate Nodes

All of the previous discussion has made the assumption that the Intermediate nodes along the path would not be compensated. This makes the initial discussion simpler, however, it is also possible to incentivise these devices as well. The way in which we propose to do this is by identifying the path packets are taking by computing a path signature. The remainder of this section outlines how to compute the path signature and how the pay channels would be involved.

We could continue to use the established pay channels between the superpeer and the devices. The pay channel would be used in the direction towards the superpeer when a device is purchasing traffic, and in the opposite direction when a device is forwarding traffic, or selling data directly. Note, the forwarding is only incentivised when the data is Internet data (internal mesh traffic still remains free).

For each source-destination pair, there may be more than one paths/routes between them. Even if only one route exists, we still have the problem of identifying which nodes participated in the route and how much data they helped in forwarding.[2] If there is more than one path for each

[2] In building the system, we considered multiple methods to solve this problem. One naïve approach considered was that whenever a packet is forwarded, the forwarder appends its MeshID in the data packet. However, two problems exist with this approach. First, since the MeshID is 160 bits, the size of data packet grows fast when going

source-destination pair, it becomes even more complicated. Thus, it is important to track the contribution of the intermediate peers.

The solution adopted by RightMesh is described in the topology below:

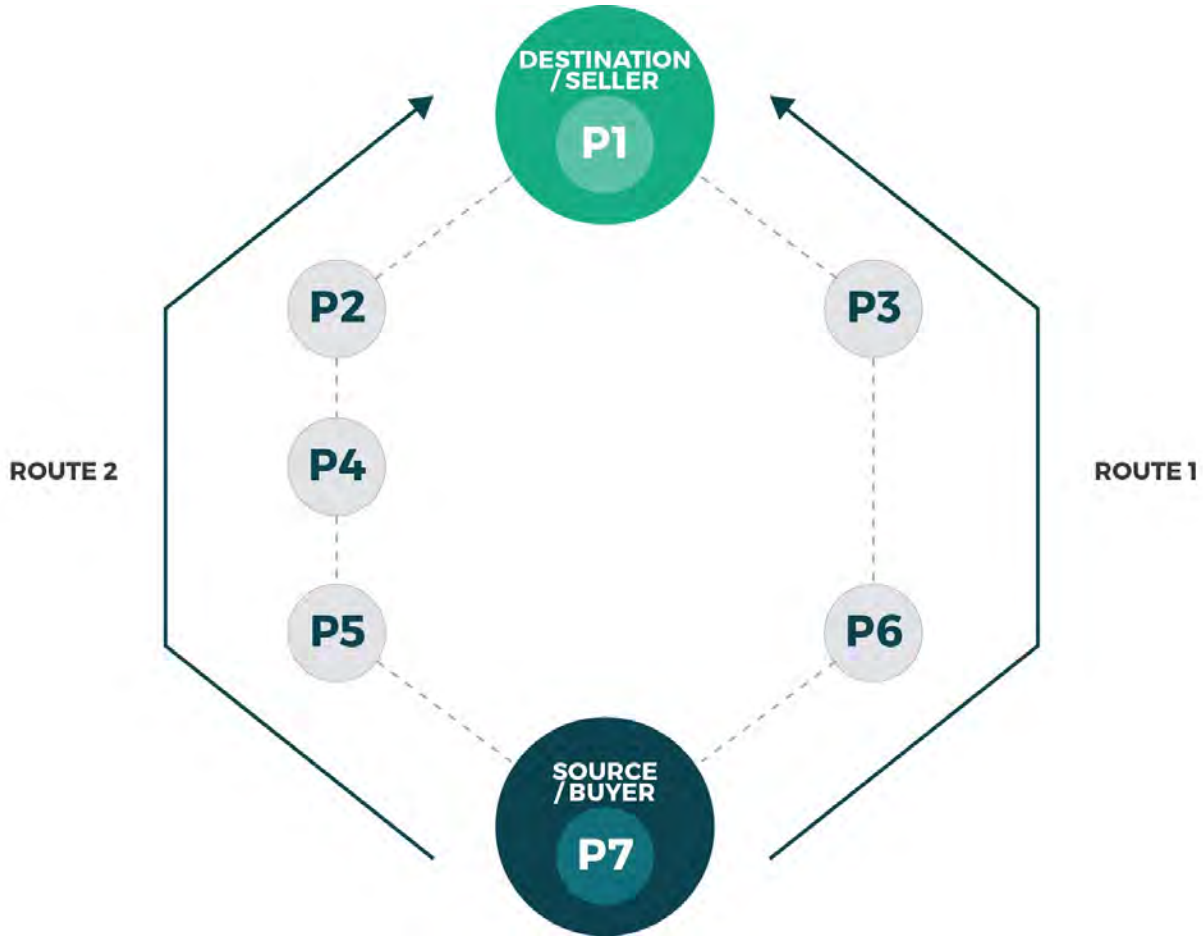

Figure 17: XOR to track intermediaries (to eventually reward them)

It has seven peers. Assume the buyer of wireless data is the dark blue peer, denoted by P_7, at the bottom of the figure, and the seller is the green peer, denoted by P_1, at the top. The buyer now

through a long-hop path. Second, we have a maximum packet size for a UDP packet in our RightMesh library, and meanwhile, we always try to deliver data with as less number of UDP packets as possible. That means adding extra MeshIDs into a data packet may exceed the maximum size of a UDP packet and an existing UDP packet may need to be split into two.

would like to send some data to the Internet via P_1. For simplicity, we use P_1, ... , P_7 to represent the MeshIDs of these peers.[3]

Assume P_7 sends out a packet, it first conducts the bitwise XOR operation between 0 and its local MeshID. That is, 0x0^P_7. The result should be still P_7. This result will be included in the packet and will be used by the rest of the downstream peers. When the packet is sent to P_1 via the left route, every intermediate node first uses the existing result contained in the packet to calculate a new result by XOR with its own MeshID. Then, it replaces the original result by the new one and saves the new one locally. For example, P_5 first calculates the result of P_7^P_5 and use it to replace the original P_7. After, it saves P_7^P_5 locally. This process repeats until the packet reaches P_1. The eventual result should be P_7^P_5^P_4^P_2^P_1. Note that, the length of this result is still 160 bits since every MeshID is 160 bits. We refer to this result as X_1. An instance of HashMap at P_1 uses X_1 as the key and save the amount of data traffic as the value of the key X_1 by tracking all the received packets tagged with X_1. Similarly, for the packets going through the right route from P_7 to P_1, the eventual result should be P_7^P_6^P_3^P_1 = X_2. Since there are 2^160 possibilities of the result, the chance of X_1 = X_2 is so small and can be counted as 0.

The keys of X_1 and X_2 together with their values will be logged into the deployed smart contract either in the final stage of the wireless data sale or in a periodic manner. Then, everyone who has X_1 or X_2 can generate some reward by sending a reward request to the smart contract with X_1 or X_2 as the argument.

For each source-destination pair, whenever an intermediate peer finds a new XOR result in the packets calculated sent from its predecessors in the upstream, the predecessor will be used as the next hop to deliver an end-to-end ACK. Note that the end-to-end ACK is sent from the destination back to the source. Inside the end-to-end ACK, we have the result of similar XOR operations from destination node to the source node. Since the XOR operation possesses the commutativity and associativity, an intermediate peer can use both the received XOR result in the end-to-end ACK packet and the previous saved result to get either X_1 or X_2 depending on which route that the intermediate peer is located at. This peer can thus directly request the reward by sending a request to the smart contract with either X_1 or X_2 as the argument. Due to the huge space of MeshIDs,

[3] A prerequisite to understanding this part is the binary Exclusive OR operation (See: https://en.wikipedia.org/wiki/Exclusive_or)

which is 2^160, it is quite unlikely that an attacker finds either X_1 or X_2 by guessing. Thus, such a system is safe and reliable.

In our system model, recall that the superpeer is used as a special caching relay between two geographically separate meshes. It will also be used in the longer term as a NAT / Proxy for general purpose Internet traffic. The superpeers will be sending out low level networking ACKs and be receiving the DATA packets to be relayed from one mesh to another. These packets contain the path signature with which a particular packet has traversed and so it is possible to keep track of an determine which devices to provide the fair share of the forwarding data towards. This will likely require some community feedback, however it will likely be capped at some fixed percentage of the data price from the seller. For instance, if the seller is selling data at 5 MESH per MB, and we set a fixed fee of 20% for infrastructure, the actual price for the buyer would be 6 MESH with 1 MESH being split equally among all devices forwarding data.

Supply of Traffic and Infrastructure

It is expected as the network grows initially, the use cases for the mesh will be primarily around sharing Internet connectivity with very localized parties. For instance, sharing data between a few friends where only one has sufficient Internet connectivity.[4] This will simply be a problem of density. As RightMesh grows in popularity, and as it is adopted in places where a user population density is greater and the problem of connectivity is much more pronounced (such as the developing world), RightMesh will start to cover wider areas. The multi-hop, multi-path, multi-technology capability will set RightMesh apart from competitors who have designed solutions which will not scale technically.

As more apps are designed for RightMesh, and as we make more partnerships with device manufacturers, app platforms, and ad distribution providers, there will be more devices with the RightMesh library running passively on a network. This will increase the value of the network, leading to even further increases in the density of devices, and enabling the creation of even more diverse applications powered by RightMesh. This will create a moat to future mesh platform developers should they emerge.

[4] RightMesh could be used to accelerate content delivery to an individual client node. Take the example of three friends each with 3G connectivity to their mobile devices. A client node may compensate the two other nearby nodes to use their Internet connectivity to have multiple concurrent connections to a content source, recombining the file across the mesh.

As RightMesh spreads in usage and is applied over wider areas, there will be many opportunities where users will be able to adjust their prices to compete with other providers and client users will have a wider variety of paths to take in the network. RightMesh can join forces with applications which aim to provide WiFi coverage maps, and instead of WiFi coverage, RightMesh could provide pricing maps. Areas with limited connectivity (or expensive coverage) would end up being places where people could physically move to so as to provide coverage for others and make a profit. An apt comparison here is to a ride-sharing driver relocating their stationary vehicle to the most optimum location to maximize their revenues.

Temporary Storage for Mobility

One of the key potentials for increasing the value of RightMesh to users is the delay tolerance which is possible and built into the platform. Data may be sent and preserved for delivery to another peer even when the peers have become separated from each other. That is, the devices may be in separate meshes, or the current mesh has split, or one or both sides lacks an Internet connection. This process means that less Internet data is required to successfully deliver intended content to the recipient. Rather than starting over in entirety, if anything has successfully been delivered, RightMesh will pick up where it left off when the path broke. Furthermore, sending will continue automatically when a new route to the intended receiver has been detected. This type of temporary storage is called “forwarding storage” and is used when a device temporarily disconnects from the network mid-data stream. Forwarding storage is used for faster recoveries during transmission.

Secondly, it is possible that client devices may also reserve some portion of their storage capacity for caching in the mesh. This not only improves performance significantly by reducing the number of requests sent directly to the Internet, it also provides a semi-offline way in which applications, advertisements, Web services, and other content (videos, music, images, maps, and other media) may be distributed. This creates many new opportunities for emerging markets that the fully-connected world already takes for granted. This type of storage is called Offline Transit Storage. With this type of storage, an entire artifact (think: file, video, music, advertisements, etc.), or potentially only a portion of it, is stored on a device indefinitely so that devices that are connected to fully offline meshes may retrieve the artifact as the mobile device moves from one offline mesh to another.

This feature of RightMesh could be greatly enhanced by supporting or collaborating with existing solutions in this space, including IPFS/Filecoin and Storj, which currently operates in the traditional infrastructure based p2p space. A collaboration here could enable these existing solutions to expand to devices and people which are unreachable with existing infrastructure. Furthermore, an IPFS hash could be sent efficiently across the mesh allowing the user to retrieve the file from the most appropriate node.

Developers API

The developers API is meant to be as simple as possible for the developer to use. There is not really any knowledge of networking that is required in order for the mesh to function. The main functionality can be summarized as follows:

- Peer discovery and updates
- Encryption key exchanges
- Sending and receiving data to a mesh peer
- Executing Ethereum transactions and contracts
- Allowing the user to configure their role in the network

All of the code is event driven. When a new device joins the mesh and it is running the same app as another device, upon discovery of each other in the library, an event is fired by our library. The developer must define a handler function for the event. In the case of a peer discovery event, the developer may wish to send a message to the new peer, update a UI element showing a list of peers, exchange encryption keys, etc.

All of our major functionality works in this manner. The sending of data is also quite simple. The developer simply calls the send function, provides the MeshID of the peer to send the data to, and the byte array of the data to be sent. The library will figure out how to split the data up, how to route the data, and how to deliver it successfully. On the other device, a received data event will be generated which can be handled by the developer with their own custom function.

The full developers API is available on our website at: https://developer.rightmesh.io/api/latest/

There is also a detailed getting started guide available at:
https://developer.rightmesh.io/reference/

End-Users

RightMesh from the End-User's perspective is designed to be as easy to use as possible. The mesh by default uses an autonomous mode which tries to form connectivity wherever it can. We have the philosophy that the library should never interrupt the way you are currently using your phone, so it will not switch off of existing Wi-Fi networks to join the mesh (it will attempt to use Bluetooth or Wi-Fi direct instead to form connections).

We also believe in giving the user the choice in how they wish to participate. If the user wishes, they can disable Internet sharing, and even choose not to participate as infrastructure. We also support an advanced mode where power users or developers may control their role in the network at a more fine grained level (selecting whether the device should setup hotspots, perform switching functions, enable / disable bluetooth, Wi-Fi and Wi-Fi direct independently, etc.).

Creating Feedback Loops

One of the biggest challenges in the mesh will be scenarios where MESH tokens are collected and concentrated and a lack of supply will circulate the mesh. First, having a minting process tied to actions that are critical to the mesh functioning will encourage people not just to speculate, but actually participate in sharing data. Second, by continuing to add more value than just data sharing, it will encourage more usage and feed into a cycle that drives more growth. Some of these initiatives include supporting ad and app distribution, supporting "data mule" behaviour, incentivizing storage/caching, processing resources -- all over the mobile mesh where competing platforms struggle to reach beyond one-hop away from the Internet.

Upon a successful token generation event, some portion of contributors will have received RightMesh Tokens which they intend to use to purchase resources from sellers on our mesh. At launch, there will likely be a mix of both contributors and users who have not yet received any MESH tokens. Users which have not yet received MESH tokens will be able to earn tokens by selling

their own data (the fastest way which includes earning MESH tokens from consumers of data), or by participating as infrastructure (where MESH tokens will be earned much more slowly).

There will be people who are attracted to the entrepreneurial route of becoming a data seller. These people may not have any MESH tokens initially, but they can accumulate tokens at a higher rate by reselling their data, or through other exchanges of value (viewing advertisements, sharing applications, being a Routing Node). RightMesh will receive tokens as a commission for data selling, relaying and verifying Ethereum transactions (while running a Superpeer), or for facilitating data distribution across the mesh. RightMesh could then perform other actions and feed the tokens back into the network. Furthermore, those who have earned tokens by being a Data Seller or by exchanging value across the mesh (storage, processing power, content creation, attention, etc.), may now choose to sell their RightMesh Tokens to people who would like to purchase more in order to buy more resources on the mesh.

RightMesh Tokens may be used by RightMesh-powered mesh applications and services. Rather than setting up their own token or cryptocurrency, and figuring out how to operate it within the confines of the mesh (including the complexities of setting up the signing procedures and infrastructure to relay requests to Ethereum or some other crypto network), anyone who develops a RightMesh mobile app or game could simply use the MESH tokens as proxy for in-app currency, to trade for in-app items, or even real-world goods and services (e.g., compensating a driver after using a decentralized ride sharing app device-to-device).

Summary of Technical Roadmap

While there are details and hints at our roadmap scattered through this document, this section collects the most important milestones and future roadmap elements to come in one spot, so that readers can understand clearly what has been built, what will be built, and what needs to be fleshed out.

What Works Today:

As of December, 2017, RightMesh has a basic mesh networking platform developed for Android devices. The platform as it stands now is a library that supports multihop connectivity between

many devices (we have had up to 20 so far on a single mesh) and can cover long distances (we have achieved about 1000 feet with only 5 devices; we will need to do further experimentation to cover longer distances).

There are several sample apps available on our public GitHub repository: https://github.com/RightMesh including a "hello mesh" app which gives an introduction to developers, a colour changing demo that illustrates how our library forms routes compared to the broadcasting libraries, a "ping test" type of app which is good for coverage testing, and soon there will be a simple hello Java app which shows the same type of code running on a something like a personal computer, Raspberry Pi, or other device running Java.

The mesh portion of the library is becoming quite mature. The connectivity aspect supports Wi-Fi, Wi-Fi Direct and Bluetooth and has two provisional patents covering it. It is partially autonomous right now, in that once the user selects which technologies to enable and which role they wish to be in (hotspot, switching, or client) the network will self form. We are working on what we call "fully autonomous" mode where the library can figure out the best mixture of these roles and technologies to enable.

We have put quite a lot of time into a Developer's Portal, so our library is easy to program with, and well documented. This includes Javadocs, a step-by-step *Getting Started Guide*, and information about what to include in Gradle.

On the crypto/incentivization side, we acknowledge that we are not experts in this area and have been seeking both talent and advice on best practices in implementation. However, we have been able to achieve a few key milestones. The first question we had was, "Can we execute Ethereum transactions from our mesh, several hops away from the Internet?" Using the Ethereum-j library, plus the Ethereum accounts we generate as our MeshIDs, we can do this using remote signed transactions. These transactions can be relayed into our superpeers which execute the transactions on behalf of the phones. You can think of the phones as specialized thin clients that can operate many hops away from the real Internet connection.

We have also made some inroads with some proof-of-concepts for contracts. We have worked with a sample token contract to represent our app token and issued a pool of tokens. We have created a contract that can act as a data pool where sellers can offer their data for sale for a price. We have also implemented the buyer contract where buyers pre-purchase their data. We've got two apps working on Android where one is a buyer and one is a seller. They are not connected to the mesh yet, but they simulate this and are connected directly to a laptop running a full Ethereum node. The buyer is able to send data by executing all of the pieces of the contracts as spelled out in this white paper. The last step is to integrate this into the mesh platform to fully realize the vision we have presented.

What is Next to Come:

The next major milestone on the mesh side is real performance analysis. While we don't have official performance numbers we can stand by yet (because we haven't obtained them over a wide number of hops using a scientifically rigorous process), we have small scale evidence (over a small number of hops with toy types of applications) that we can achieve high performance (on the order of Mbps). As we go farther down this road, we will release further apps that allow for throughput and delay measurements, so the community can try out the mesh and perform their own tests to see where and how it works best.

Along with the performance analysis is getting a clear idea on how the network performance changes as the network grows. Where are the bottlenecks, how does performance degrade over hops, and how many Internet sharing devices can one superpeer support? How many consumers can one internet sharing device support? Also, with <x> devices consuming data, how quickly will <y> data be consumed? After we answer some of these questions, there will be lots of opportunities to improve the performance as we will learn a significant amount about where the network is weak, while helping us generate ideas on how to improve.

One of the biggest challenges for us on the network side is automated testing of the library. Given that much of the operation of the library depends on complex connection topologies between groups of phones, it is difficult to simulate effectively. There is some portion of the library that we have applied unit testing and limited offline testing to; however, some of it requires real device testing as there are aspects that need to be tested specific to particular Android versions, the

orientation of the phones near to each other, and the order in which the devices discover each other. We would like to support phones from Android 4.x to present; however, with the wide range of capabilities, the way Android handles permissions, and other differences between devices, this means there is a huge scaling problem to automating the whole process since the number of testing possibilities grows exponentially as the mesh size grows. We are looking at ways to automate as a much of this as possible so future builds are stable across a wide range of devices.

As mentioned on the incentivization side, the biggest challenge going forward is integrating the contracts with the mesh. We don't anticipate this being particularly difficult because we have already had the mesh execute remote signed transactions, and we can do the same with contracts. The biggest challenge here is getting the sidechain part right: whether this is using Ethereum, Lightning, Ripple, or some other technology. In addition, getting the economics of the system correct so that there is stability, so that tokens don't end up getting stuck and not used in the network, etc. This is where we are looking for the most community feedback and advice as we are more mesh focused than blockchain focused, but we recognize that blockchain technology is the best path toward incentivizing the mesh.

Once the basic data sharing incentive structure is in place and well understood, the next step will be apps for ad and app distribution, "data mule" incentives, storage/caching, processing, and more.

We are also rapidly growing our team in Canada, and are on the lookout for more talent. Whereas others are building walls, do note that Canada has a new two-week processing program for skilled workers under a new initiative to invite the world. We do not discriminate based on location, nationality, religion, or sexual orientation, and are consistently recognized as one of the Best Workplaces in British Columbia, including winning the award in two straight year from the BC Tech Association as the best tech company in BC that balances, "Work, Life, and Play". So if you have experience (either academic or professional), passion, and a commitment to doing things right, we would appreciate hearing from you.

We welcome feedback via Git, Telegram, and more methods identified below.

Additional Resources & Links

- RightMesh White Paper: https://www.rightmesh.com/whitepaper
- Website: https://www.rightmesh.io
- Developer Portal: https://www.rightmesh.io/developers/
- Corporate Blog: https://www.rightmesh.io/news/
- Twitter: https://twitter.com/Right_Mesh
- Telegram: https://telegram.me/RightMesh
- GitHub: https://github.com/rightmesh
- Careers: https://www.left.io/careers

Legal & Regulatory

The RightMesh team is aware of potential risks associated with a decentralized mesh networking platform and the associated tokens. As of the date of publication of this White Paper, RightMesh Tokens have no known potential uses outside of the RightMesh platform ecosystem and are not permitted to be sold or otherwise traded on third-party exchanges. This White Paper does not constitute advice nor a recommendation by RightMesh, its officers, directors, managers, employees, agents, advisors or contractors, or any other person to the recipient of this White Paper on the merits of participation in the public contribution offering. Participation in the public contribution offering carries substantial risk and may involve special risks that could lead to a loss of all or a substantial portion of such contributions. Do not participate in the public contribution offering unless you are prepared to lose the entire amount you allocated to contributing in exchange for RightMesh tokens.

RightMesh tokens should not be acquired for speculative or investment purposes with the expectation of making a profit or immediate re-sale. No promises of future performance or value are or will be made with respect to RightMesh tokens, including no promise of inherent value, no promise of continuing payments, and no guarantee that Rightmesh tokens will hold any particular value. Unless prospective participants fully understand and accept the nature of RightMesh and the potential risks inherent in RightMesh tokens, they should not participate in the public contribution offering. RightMesh tokens are not being structured or sold as securities. RightMesh tokens do not provide participation in RightMesh GmbH or its parent company Left of the Dot Media Inc. and RightMesh tokens hold no rights in RightMesh.

RightMesh tokens are rewarded as a functional good and all proceeds received by RIghtMesh may be spent freely by RightMesh absent any conditions, save as set forth in this White Paper. This White Paper is not a prospectus or disclosure document and is not an offer to sell, nor the solicitation of any offer to buy any investment or financial instrument in any jurisdiction and should not be treated or relied upon as one.

Holders of RightMesh tokens assume risk when participating in trading activities and such activities have inherent risks. Unforeseen problems could result in the loss of all of a RightMesh token holder's funds RightMesh token value.

All information here that is forward looking is speculative in nature and may change in response to numerous outside forces, including technological innovations, regulatory factors, and/or currency fluctuations, including but not limited to the market value of cryptocurrencies.

CAUTION REGARDING FORWARD LOOKING STATEMENTS

This White Paper contains forward looking statements or information (collectively "forward-looking statements") that relate to RightMesh's current expectations and views of future events. In some cases these statements can be identified by words or phrases such as "may" / "will" / "expect" / "intend" / "plan" / "believe" / "potential" / "is/are likely to" / "continue" or the negative of these terms or other similar expressions intended to identify forward looking statements. RightMesh has based these forward-looking statements on its current expectations and projections about future events that it believes may affect its financial condition, results of operations, business strategy, financial needs, or the results of a token generating event or the value or price stability of RightMesh.

In addition to statements relating to the matters set out here, this White Paper contains forward-looking statements related to RightMesh's proposed technology and operating model; such statements speak to RightMesh's objectives only, and is not a forecast, projection, or prediction of future results of operation.

Forward-looking statements are based on certain assumptions and analysis made by RightMesh in light of its experience and perception of historical trends, current conditions and expected future developments, and other factors it believes are appropriate, and are subject to risks and uncertainties. Although the forward-looking statements contained in this white paper are based upon what RightMesh believes are reasonable assumptions, these risks, uncertainties, assumptions and other factors could cause RightMesh's actual results, performance, achievements, and experience to differ materially from its expectations expressed, implied or perceived in forward-looking statements. Given such risks, prospective participants in this token generating event should not place undue reliance on these forward-looking statements.