

Althea whitepaper

Jehan Tremback, Justin Kilpatrick, Deborah Simpier, Ben Wang

- v0.5 May 2017
- v1 May 2018
- v1.1 Oct 2018
- v1.5 April 2019

As the number of connected individuals and devices expands, the “last mile” continues to be the greatest challenge both in the connected and developing worlds, representing a disproportionate portion of the cost and difficulty of connecting the world (“Last mile” refers to the distance between an internet exchange and a user, often less than 10 miles).

Althea is meant to operate on the last mile, from a source of internet connectivity (such as an internet exchange or tier-1 or -2 network connection, or even a business grade ISP connection) to the end user, and creates a decentralized ISP. The last mile is currently an inefficient market and many areas only have one ISP

(http://transition.fcc.gov/Daily_Releases/Daily_Business/2017/db0503/DOC-344499A1.pdf).

Althea aims to replace centralized ISPs with a competitive market of individuals and businesses participating in one decentralized network.

Althea’s goal is for any person to be able to install a piece of networking equipment (such as a long range radio transmitter), participate in the decentralized ISP, and receive payment for the service.

- Switching costs within the system are reduced, as nodes switch peers to find a route with the best combination of reliability, bandwidth, and low cost.
- Advertising and marketing costs for people running nodes are eliminated, as the only advertisements in this system are the automatic advertisements of price and route quality between nodes. This makes things easy for new entrants.
- Contract and billing costs are eliminated by an automatic node to node billing and payment system using blockchain payments.

1. Overview

Althea allows routers to pay each other for bandwidth using cryptocurrency. An important architectural detail is that nodes only pay neighbors for forwarding packets. On top of this pay-for-forward network, we build a system allowing consumers to pay for internet access. Althea is intended to be used in local “mesh” (<https://www.cs.columbia.edu/~vpk/papers/wcn.commag11.pdf>) networks.

Network overview

To simplify the explanation of Althea’s network architecture we present several roles that devices may perform. Note that these are logical roles, not physical ones. So a single device may perform several or even all of these roles simultaneously:

- User nodes are installed by people who want to buy internet access on Althea. You can think of a user node as being similar to the router and/or modem that is installed by a traditional ISP. The difference is that it is independent of any one ISP. User nodes bridge traffic from non-Althea devices into the Althea network by providing Wifi hotspots and/or wired LAN ports.
- Relay nodes are installed by people who want to earn money by forwarding internet traffic. These will often be more powerful and may be placed in advantageous locations with good line of sight to other nodes. Note that many nodes will be acting as both relay nodes and user nodes. The software to buy and sell bandwidth as a user node is necessary and sufficient to be an relay node.

- **Gateway nodes are relay nodes**, but they are also connected to a source of cheap internet bandwidth such as an internet exchange, an internet backbone connection, or even a business-grade connection from a conventional ISP. They act as a connection from Althea's physical layer to the outside internet. However, they are shielded from having to take legal responsibility for traffic on the network by the exit nodes.
- Exit nodes are not necessarily part of the local physical network, but can be hosted in a datacenter reachable over the internet. They are connected to gateway nodes over VPN tunnels. Exit nodes provide an endpoint to verify quality metrics propagated by nodes in the network. This enables automatic selection of gateway nodes by the routing protocol. Exit nodes also take on some of the roles of an ISP, performing network address translation to route requests onto the public internet and dealing with copyright complaints etc. This allows gateway nodes to act as pure providers of bandwidth, without having to take on any legal risk resulting from the use of their service.

Read more about the network architecture in [Network](#).

Routing overview

In [Routing](#), we define a couple of extensions to the Babel routing protocol. Babel was selected because it has several useful properties for our purpose. However, any distance vector routing protocol could be modified to exhibit the properties Althea requires. Distance vector protocols are already used extensively on the internet. One well-known distance vector protocol is [BGP](#) (<https://tools.ietf.org/html/rfc4271>).

Routing in distance vector protocols is based on an advertised connection quality metric. Nodes send an announcement packet stating their identity and existence to the network once every predetermined period. These announcement packets are then passed from node to node. Each node updates the metric to reflect the connection quality between it and the neighbor it got the announcement from. Using this information, **each node is able to build up a routing table of the best neighbors to forward packets in order to reach any destination on the network in only $O(n)$ time on each node.**

We propose two main additions to distance vector routing:

- **A verifiable quality metric.**
- **A price metric.**

A verifiable quality metric is a connection quality metric that can be verified by a node and the destination that it is sending packets to. Our first extension to Babel allows nodes to verify the metrics advertised by their neighbors.

To advertise prices a second metric is added to the routing advertisements, this time containing a 'price' value for some arbitrary but agreed upon amount of data transfer. When passing advertisements each node updates the price field with their bid for passing data. Routes are then selected by optimizing the quality metric vs the price metric and paying the selected the full sum required to route all the way to the destination.

Metering overview

Nodes keep track of data they have forwarded for their neighbors, and how much they have been paid. If these two amounts do not match up, they must have some way of cutting off access to the **delinquent neighbor**. Blocking the neighbor's MAC address could be one way to accomplish this, but MAC addresses are easily changed and spoofed. Similarly, exit nodes must be able to protect and identify traffic from user nodes.

In [Authentication](#), we cover the use of tunnels to allow neighbors to verify traffic between one another, as well as allowing exit nodes to authenticate traffic from user nodes.

Blockchain overview

Althea uses a proof of stake blockchain built on the Cosmos platform. This blockchain is maintained by validators running the Althea blockchain software. These validators are given power to validate the Althea blockchain by holders of the Althea governance tokens, who delegate tokens to the validator(s) of their choice. Validators who perform this role incorrectly or have too much downtime are "slashed"- some of the tokens delegated to them are burned. More about this system in the [Cosmos whitepaper](#) (<https://github.com/cosmos/cosmos/blob/master/WHITEPAPER.md>).

Payments are done on this blockchain using stablecoins, which maintain their price relative to national currencies. Validators and those delegating Althea governance tokens to them earn transaction fees from each payment. Initially the stablecoin used will be Dai, which is pegged to the US dollar.

This blockchain will also be used as a platform to run subnet DAOs, which allow local subnet organizations to administer parts of the Althea network.

2. Implementation

Routing

Routing in Althea is based on the Babel routing protocol (<https://tools.ietf.org/html/rfc6126>). Babel is a distance vector protocol which has proven to be robust and performant. All distance vector protocols are based on a distributed form of the Bellman-Ford (https://en.wikipedia.org/wiki/Bellman-Ford_algorithm) pathfinding algorithm. Nodes first perform some kind of link quality test on the connections to their neighbors. This is known as the “link cost”. They then share information about which destinations they can reach at which quality (this starts out being only their immediate neighbors).

Whenever a node receives information about a destination, it combines this information with the link cost of the neighbor it received this information from (we will use the word “destination” to refer to the destination of data packets. Babel refers to this as “source”, because it is the source of routing packets). This composite score is known as the “route metric”, and represents the quality with which the destination can be reached across several hops. The neighbor offering the best metric for a given destination is selected as the next hop. All packets being sent to the destination will go through this neighbor. Babel implements this selection by adding and remove routes from the Linux kernel routing table.

From the Babel specification:

As many routing algorithms, Babel computes costs of links between any two neighbouring nodes, abstract values attached to the edges between two nodes.

[..]

Given a route between any two nodes, the metric of the route is the sum of the costs of all the edges along the route. The goal of the routing algorithm is to compute, for every source S, the tree of the routes of lowest metric to S.

Route metric verification

All current distance vector protocols, including Babel, have a major weakness. All information about link cost and route metrics is provided on a completely trusted and unverified basis. There is nothing stopping any node from claiming that it has the best route to any destination. This is usually not a problem, since most networks today are owned by one entity. In Althea, nodes are owned by many people and entities, all competing to provide the best service. Leaving this vulnerability unaddressed would allow financially-motivated attacks, such as nodes claiming to have better routes than they actually do in an effort to get more business.

We propose a modification to the Babel protocol to allow for verification of routes. Our modification involves the addition of a “verifiable metric”. We define a “verifiable metric” as a distance vector metric that remains the same when observed end-to-end or by individual nodes. For example round trip time (rtt) as a metric can be summed along the route or computed purely by the source and destination. Using a verifiable metric user nodes and exit nodes can get an independent sample of the connection metric over an encrypted tunnel between them. The tunnel keeps nodes from cheating by prioritizing route verification packets. This technique is known as stealth probing (<ftp://ftp.cs.princeton.edu/techreports/2005/730.pdf>).

The metric calculated is expected to be close to the overall metric advertised for the destination by the neighbor currently forwarding packets to the destination. This gives us a way to verify the accuracy of advertised routes. If the metric does not match the metric advertised by the selected neighbor, the neighbor’s accuracy score is affected, and the metric through this neighbor is adjusted.

Verification scheduling

We have the ability to test and verify the routes advertised by neighbors, but we need some way of deciding which routes to verify. Verification runs on a timer. Each verification cycle a node follows this procedure to choose a route *r* to verify:

- Start with the set of all destinations that have been used in the past *x* seconds. *x* will be set at a reasonable constant, or it could be adjusted to control verification focus.
- Select a destination *d* from this set at random.
- Get the set of all routes that are feasible and have a destination of *d*.
- Select a route *r* from this set at random.

Accuracy score

When we get a new verified route metric, we update the neighbor’s accuracy score *s* with the following procedure:

```
d = destination
n = neighbor
c = route cost over tunnel to d
m = route metric advertised for d by n
A = accuracy scores of n, indexed by destination
S = lowest allowed average accuracy score

a = minimum(m/c, 1)
A[d] = (A[d] + a) / 2
s = average(A)

if s < S
    n's routes removed from routing table
```

A per-destination accuracy score *a* is calculated as the proportion of the verified route cost *c* to the route metric *m* advertised by the neighbor *n*. It is then averaged with the current accuracy score for that destination *A[d]*, and *A[d]* is set to the new value. We then take *s*, the average of all active destination accuracy scores for the neighbor. If *s* drops below some adjustable value *S*, the neighbor’s routes are taken off of the kernel routing table.

Route metric adjustment

The above system is used to weed out chronically inaccurate neighbors, but it also supplies us with a stream of correct route metrics. We can use these metrics to improve our routing table even before a given inaccurate neighbor is dropped. When we receive the route cost *c* above, we can start using it instead of the neighbor’s advertised metric when selecting routes. We continue using *c* for a duration *D*.

How long to make *D*? If *D* is too short, then nodes will go right back to trusting an inaccurate metric that they recently had to correct. If *D* is too long, then nodes will miss out on legitimate updates about newly improved routes.

To strike the right balance, exponentially increasing time-out will be used for bandwidth corrections beyond a small tolerance. A node that has participated in a correction will calculate the duration *d*, how long ago it last performed a correction for a given route and the size of the correction *s*. When participating in another correction on the same route the duration *D* to apply the bandwidth correction will be determined as:

D = (C1 / d) ^ C2

Where *c1* and *c2* are constants to be adjusted and hardcoded. This formula will make it take longer to go back to using a neighbor’s advertised metrics if the advertised metrics needed to be corrected recently, and/or required a large correction.

Price metric

We also need a way to propagate a price for each route, and take this price into account when making forwarding decisions. Babel already includes a mechanism for adding arbitrary “External Sources of Willingness”. This works by having nodes add a number to the metric they have

calculated for a route. This doesn't work for us for two reasons:

- First, the route metric in our system is verified. Modifying it arbitrarily would break this verification.
- Second, the price must be distinct from the route metric, because it will be used to determine payment amounts. For these reasons, we use a separate price metric.

The requirements of this price metric are simple. It consists of an additional 16 bit price field in each Babel update TLV and in each route in a node's routing table.

As update packets are propagated through the network, each node increases the route's price by a certain amount. The simplest way to determine how much to add to the route price is with a constant set by the node's operator. However, there could be many different types of automated price-setting algorithms to adjust the price based on demand or competition.

Babel's route selection procedure is extended to take this price field into account. Instead of selecting routes based purely on route metric, an extended metric m' is calculated with

$$m' = n * \log_2(m) + \log_2(p)$$

Where m is the route metric, p is the price, and n is a constant multiplier. Routes are then chosen based on m' . The log function is chosen for normalization, such that increasing or decreasing the metric or price will change the extended metric by a constant amount, no matter their absolute values. By adjusting n , nodes can determine how much weight to give price in the calculation. In other words a node will switch to a connection which is $2*n$ times more expensive per byte if its metric is 50% lower.

It would even be possible to populate multiple routing tables with routes selected at different values of n , and propagate routes from these tables under different router IDs. This or a similar mechanism could be used to allow neighbors to choose from among a range of price-quality tradeoffs.

Payments

We've chosen for nodes to pay after receiving service. This is because if nodes pay before, it would be possible for a malicious node to repeatedly accept payment and not provide service (possibly switching identities each time). They could save up their ill-gotten gains and make money with this strategy. On the other hand, if nodes pay after receiving service, malicious nodes that repeatedly receive service and don't pay will get nothing but a bad connection (you can't save up stolen bandwidth).

Authentication

We've set up a system where nodes are able to pay for traffic, but what happens if they don't pay? There needs to be some control over which neighbors receive internet access. It's easy to spoof a MAC address, so we need some kind of cryptographic authentication. One way to do cryptographic authentication of packets on radio is WPA (<https://ieeexplore.ieee.org/document/4248378/>), but we need something that can be done on wired links too. The best solution for now is to use an encrypted and authenticated tunneling software like Wireguard (<https://www.wireguard.io/papers/wireguard.pdf>). A small optimization would be to include authentication information in an IPv6 header extension, instead of encapsulation in tunnel packets with Wireguard. However, Wireguard is already highly optimized, so this is an adequate solution for now.

Nodes create tunnels with each of their neighbors and can allow, block, or shape traffic on each tunnel, depending on payment. Wireguard creates a virtual interface for each tunnel. The virtual interfaces created by the tunnels of all the neighbors on one physical interface are bridged together into another virtual interface for Babel to run on. This preserves some per-interface optimizations made by Babel.

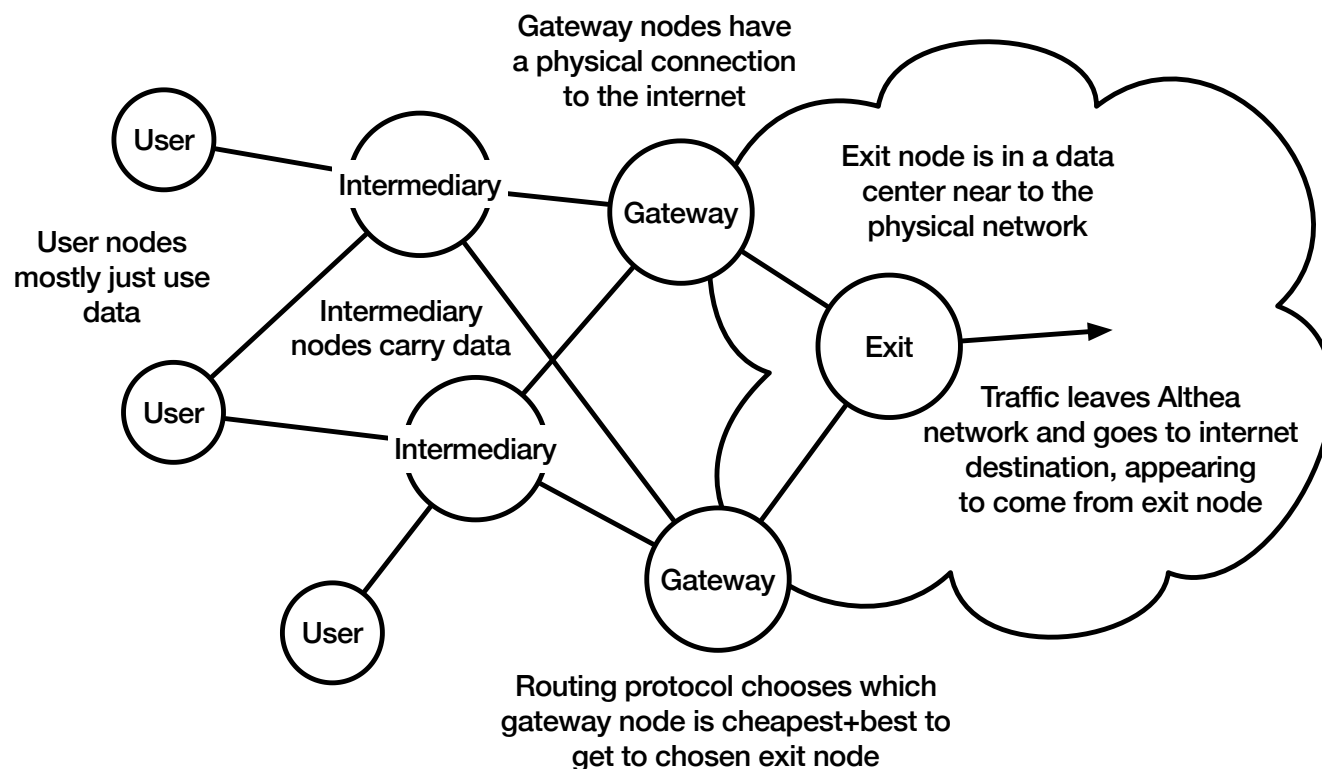
If payment from a neighbor stops, their packets are blocked by the firewall and are no longer forwarded.

Metering from the exit node to the user node is accomplished in the same way. There is also a Wireguard tunnel from the exit node to the user node for other reasons, namely privacy and as a way to use IPv6 on the mesh network, similar to [Lightweight 4over6](#)

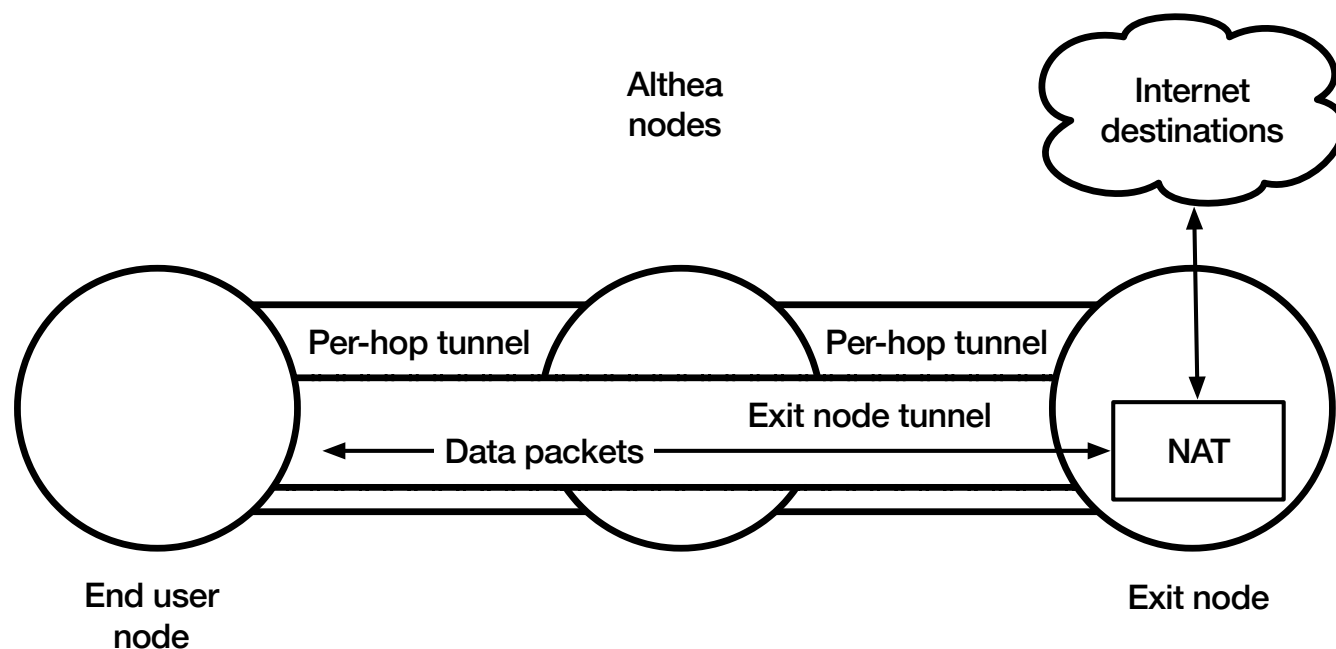
(<https://tools.ietf.org/html/rfc7596>). This tunnel provides a good control point for the exit node to cut service to the user node in the case of non-payment.

Network

The base primitive that Althea is built on is a pay-for-forward network. If all the mechanisms in the preceding sections work correctly, we have a network where nodes can pay each other very granularly for the service of forwarding data, and verify that the forwarding is happening correctly. This section deals with using such a network to provide the one of the most popular network services, internet access.



For a network to provide internet access, there must be at least one node that has a connection to the internet. Hopefully, there will be many. We call such a node a gateway node. A gateway node connects to an exit node over a tunnel. The tunnel creates a virtual interface which Babel is run on, treating it like any other link. The gateway node/exit node topology (https://sudoroom.org/wiki/Mesh/Network_topology) is used by WLAN Slovenija (<https://wlan-si.net>), PeoplesOpen.net (<https://peoplesopen.net>), and other community mesh networks.



Babel routes to destinations over tunnel connections just as well as it does over real connections. This means that the nodes do not have to do any kind of explicit gateway selection. Gateway nodes set a price and receive payment for routes to the exit node just like any other route. User nodes are connected to chosen exit nodes over encrypted tunnels, and receive internet access over these tunnels. The only thing that gateway and relay nodes see is encrypted traffic between user nodes and exit nodes.

Exit nodes

Exit nodes perform almost all the functions of an ISP, except for actually carrying packets. This lets the other nodes in the network focus only on connectivity, while exit nodes get paid for interfacing Althea to the rest of the internet, and to the business and legal worlds. Exit nodes fuse Althea's

pseudonymous, trustless, cyptocurrency powered physical layer with our current internet and society. People who are good at providing connectivity can focus on providing connectivity, while exit nodes deal with everything else.

Exit nodes:

Deal with public IP addresses: Exit nodes have public IP addresses and use them to route traffic for connected user nodes to and from the internet. They either perform NAT for user nodes or provision them with public IP addresses if the user nodes are performing NAT themselves (such as in schemes like Lightweight 4over6 (<https://tools.ietf.org/html/rfc7596>)).

Provide encrypted tunnels: All traffic between user nodes and exit nodes is encapsulated in a tunnel and encrypted. This means that users of a Althea network only have to trust the exit node with their browsing history and unencrypted traffic. Neighbors who are routing traffic for them will only see encrypted packets going to an exit node.

Verify routes: Our extensions to Babel allow nodes to verify routes between themselves and a destination. Exit nodes are the destination for a user node's outbound traffic, and the user nodes are destinations for the return traffic. User nodes and exit nodes work together to keep the nodes on the Althea network between them accurate. User nodes are implicitly trusting exit nodes to perform route verification accurately.

Deal with legal considerations: We want it to be as easy as possible for someone to set up a gateway node. Part of this is relieving them of any legal worries related to the use of their connection. Any legal complaints related to the use of an Althea network can only be directed the exit nodes, as they are the only ones routing traffic onto the internet and making it visible to the world. Gateway and relay nodes only ever see encrypted packets.

Pay for return traffic: User nodes pay their neighbors to forward traffic to the exit node they are using and onto the internet, but someone needs to pay for the traffic coming back. User nodes send exit nodes some money which the exit nodes use to pay their neighbors for the return traffic.

Subnet DAOs

Althea subnet DAOs define local Althea networks. They are lists of Althea nodes, identified by an IP address and a blockchain address. They are meant to be maintained by local groups of Althea users. We refer to such groups as “subnet organizations”. The subnet DAO defined here is the software they use. An Althea subnet DAO has two types of participants:

- Nodes, which are identified by an IP address and a blockchain address. Nodes on a given subnet connect only to other nodes on that subnet.
- Organizers, who have the ability to vote on adding and removing nodes and other organizers.

Subnet DAOs have two main purposes:

- Reward people who take an active role in promoting the use of Althea in an area and make sure everything is running smoothly.
- Provide a mechanism for these people to control membership of a network by adding and removing nodes to mitigate attacks and bad behavior by nodes in the network.

Each node on a subnet pays a continuous (per block) fee to its subnet DAO, in addition to the payments for bandwidth it is making to its neighbors. This continuous fee is mediated through the subnet fee escrow module.

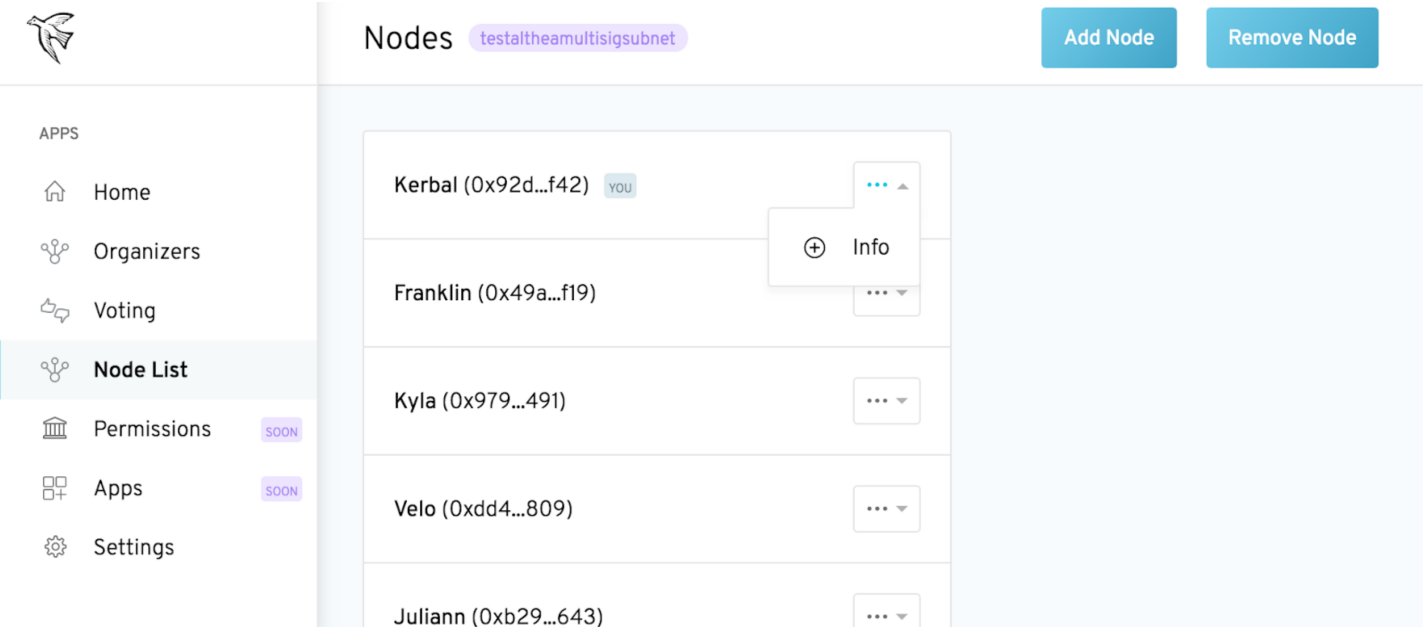
Nodes on a subnet check their subnet DAO on the blockchain before connecting to other nodes. If a node is not on the same subnet, they do not connect.

Nodes will most likely be added to a specific subnet because the organizers of that subnet DAO have introduced the owner of the node to Althea (and possibly installed their node). Subnet organizations are incentivized to add more happy customers to the system to receive the subnet fees. This dynamic is intended to incentivize the spread of Althea, by rewarding the people who help spread it, even if they don't own equipment earning bandwidth payments.

Subnet DAO implementation

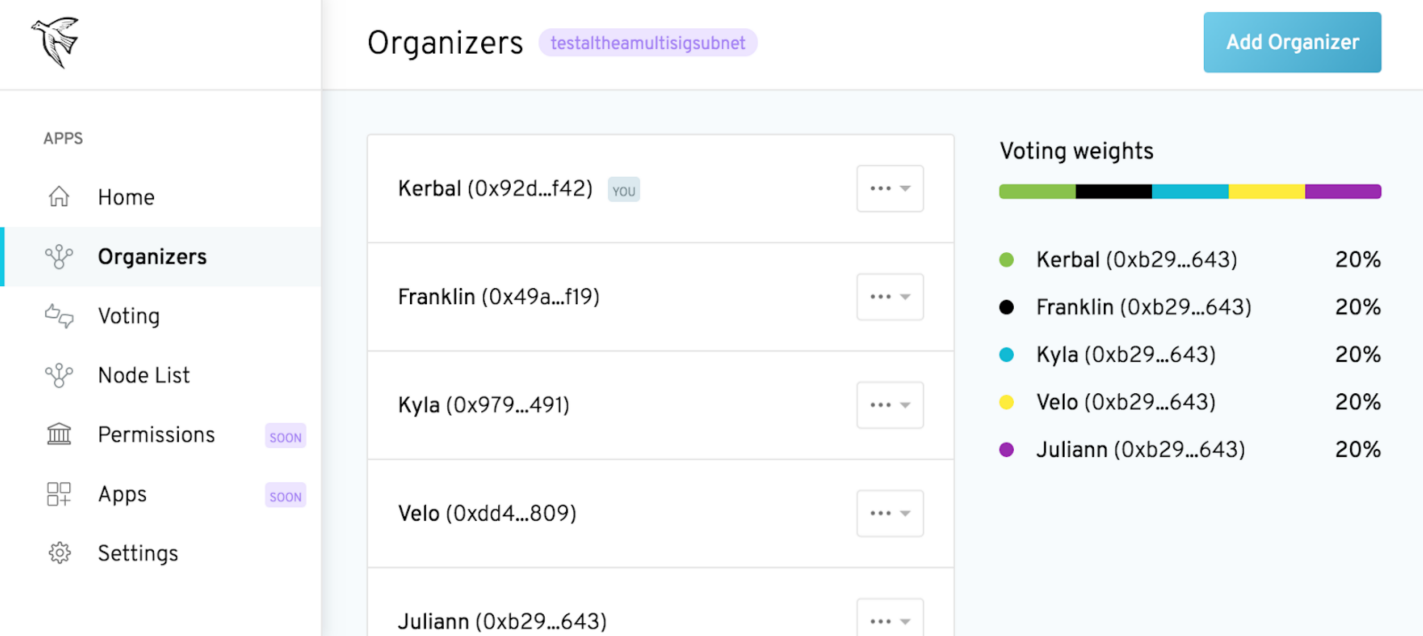
If someone wants to start an Althea network in their area, they download the Althea subnet DAO dApp, or visit a hosted version. A wizard takes them through the subnet DAO setup process. These illustrations are of a proof of concept built on the Aragon DAO framework.

NODE LIST



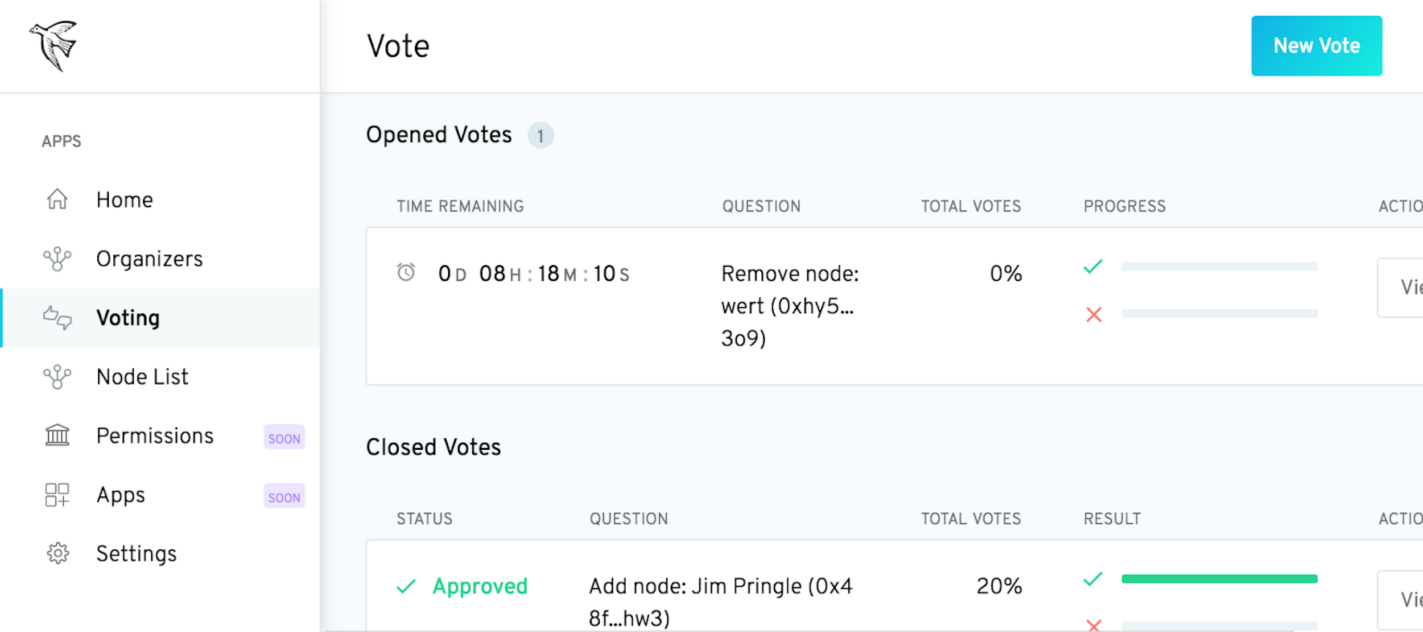
Routers on the network can be viewed on the “Node List” screen. Nodes are identified by a nickname and blockchain address, with more info such as IP address and notes available in a menu.

ORGANIZERS



Organizers are visible on another screen. These are the people who make decisions about the subnet DAO.

VOTING



Adding or removing a node or an organizer, or doing other actions like disbursing funds collected from renewal fees triggers a vote. If only one signature is required for a given operation, the vote is automatically approved.

By default, one signature is required to add a node, and two signatures are required to remove a node. A majority is required to add or remove an organizer.

Renewal fee escrow module

This module allows subnet DAOs to charge a per-block fee without the inconvenience or transaction fees of actually paying it every block. At the same time, it avoids the need for a node to trust a subnet DAO with its money. Althea nodes add some money to this escrow module ahead of time. When a node joins a subnet or leaves a subnet it calls the escrow module as part of that transaction. Subnet organizers can call the escrow module at any time to withdraw the money owed them.

WHEN A NODE JOINS A SUBNET DAO:

- Record that it has joined the subnet DAO, and the subnet DAO's renewal fee at the time it joined.

WHEN A SUBNET ORGANIZER CALLS THIS ESCROW MODULE TO GET MONEY THAT IS OWED TO THE SUBNET DAO:

- Get a list of all nodes which are currently on the subnet DAO.
- For each of these accounts:
 - Figure out how much it owes the subnet DAO, based on the renewal fee, the last block it paid the subnet DAO, the block it joined the subnet DAO, or the block it left the subnet DAO.
 - Calculate how much of the renewal fee goes to the Althea validators and curators and transfer it.
 - Transfer the remaining amount from the node's escrow to the subnet DAO.

WHEN A NODE LEAVES A SUBNET:

- Figure out how much it owes the subnet DAO, based on the renewal fee, the last block it paid the subnet DAO, the block it joined the subnet DAO, or the block it left the subnet DAO.
- Transfer this amount from the node's escrow to the subnet DAO.
- Remove the node from the list of nodes which are on the subnet DAO.

Actors

This is a list of the different participants in this system and their economic and governance activities.

Subnet organizers

Subnet organizers promote and support the Althea network in their area. They control a subnet DAO and IP range. Since the subnet DAO receives a fee from each node using one of its IP addresses, organizers stand to gain from increased adoption of the service, even if they do not own a profitable relay node themselves. Subnet organizers are also responsible for dealing with attackers on the network by revoking their IP addresses.

ECONOMIC ACTIVITY

- Subnet organizers control a subnet DAO which receives IP address renewal fees from end user and relay/gateway nodes.
- Spend money doing things to promote Althea, and their subnet.

GOVERNANCE ACTIVITY

- Add new nodes to their subnet
- Remove attackers from the subnet.

End users

End users are using the network, and while they may be excited about the concept of incentivized mesh, they mostly just want it to work. They are receiving access from one or more relay nodes that they are within line of sight of, or have another kind of connection with (Ethernet, coaxial cable, DSL, fiber, etc). They were probably sold on the concept of incentivized mesh by the subnet organizers of their subnet.

ECONOMIC ACTIVITY

- Pay relay nodes for bandwidth.
- Pay IP address renewal fees to subnet DAO.

GOVERNANCE ACTIVITY

- Are allocated an IP address by subnet DAO organizers.

Relay node owners

Relay node owners own a node which connects end user nodes or other relay nodes to the network. They make the system work on a basic level and receive the payments for bandwidth which make up the bulk of economic activity in the system. The functioning of relay and end user nodes is what makes up most of the Althea [network paper](https://altheamesh.com/network-paper) (<https://altheamesh.com/network-paper>).

ECONOMIC ACTIVITY

- Sell bandwidth to other Althea nodes.
- Either:
 - Buy bandwidth from another Althea node.
 - Pay for or otherwise procure bandwidth to the Internet in the conventional system.
- Pay IP address renewal fee to subnet DAO.

GOVERNANCE ACTIVITY

- Allocated an IP address by subnet DAO organizers.
- Have a vested interest in the success of incentivized mesh in their area, but not necessarily any particular subnet DAO.
- May have IP addresses on several subnet DAOs in order to connect to end user nodes on those DAOs.

Althea validators





ALGT holders can stake tokens to participate in blockchain validation and receive blockchain transaction fees.

ECONOMIC ACTIVITY

- If their tokens are staked on validation, validators receive transaction fees and block rewards.
- If their tokens are not staked, they will lose value from inflation at the rate of 7% per year.

GOVERNANCE ACTIVITIES

- Vote on proposals to modify and improve the Althea blockchain.

			
About	Learn	Connect	Resources
How it works	Blog	Chat	Github
Case Studies	Whitepaper	Twitter	Documentation
About the project	Youtube	Reddit	Getting Started Guide
Team		Send us an Email	Glossary
Careers			