Substratum Network

Technical Paper version 1.0

Mission

Substratum is creating an open-source foundation for a decentralized web which will provide unrestricted access to content and sharing of information for users across the globe.

Our mission is to bring forth the free and fair internet of the future by combining proven technological building blocks with emergent technologies in an innovative and holistic way to help solve many of the problems that plague the modern internet.

The Substratum Network

The Substratum Network is an open-source network that allows anyone to allocate spare computing resources to make the internet a free and fair place for the entire world. It is a worldwide collection of nodes that securely delivers content without the need of a VPN or Tor.

Because there's no single authority delivering or monitoring content, censorship and geo-restricted sites won't be an issue on Substratum Network. It doesn't matter where you live or what content you're accessing, everyone in the world sees the exact same content.

The concept for the Substratum network first originated as a closed system that would only provide content within its private network. By leveraging computer resources and decentralizing content it would enable several hosts to render sites across several hosts. What became difficult was the time and scale it would take to move so much content into this private network. On inception the network would not even provide 1% of the world’s website content. The alternate solution was to find a way to provide content in a way that could still travel peer to peer, would be secure, and would allow access to all the internet has to offer.

Substratum Node

A Substratum Node is the foundation of The Substratum Network.

It is what the average user runs to earn SUB and dedicate some of their computers' resources towards the network. People who run a Substratum Node can be rewarded with cryptocurrency for each time they serve content.

The purpose of Substratum Node is to mask and un-mask CORES packages for clandestine routing.

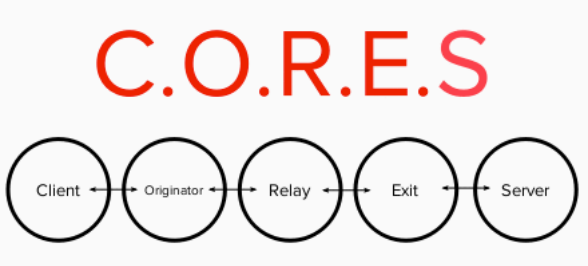
Substratum Nodes work together to relay CORES packages and content on the network. When a user requests a site, nodes use algorithms to find the most expedient and secure way to get the information to that user. Multiple nodes work to fulfill a single request in order to maintain a necessary level of anonymity.

Understanding the Components

Substratum Node and The Substratum Network contain several components and features that help provide all of the functionality. Before explaining the step by step process of how the network communicates let's understand each component of the entire network.

CORES Packages

A CORE(s) package is a series of instructions bundled in a way that allows each node within the network to receive just enough information to make a decision on how to handle a request. CORE(s) packages contain layers of requests that inform nodes how to perform one of three actions. Create the request, pass the request along, process the request.



Client: This is the device that is initializing the request such as a browser, or a video game.

Originator: The identity of the originating request node.

Relay: Identifies Nodes that will be used to hop along routes to get to its destination.

Exit: The exit node will make the true web request on behalf of the originator and send it back along the route stack.

Server: Information about the server that is hosting the requested information.

Substratum DNS

The purpose of our DNS is to be the OS-agnostic tool that the Substratum Node user interface will use to examine and manage the DNS settings of a user's machine for routing its TCP communications over the Substratum Network. It is built as a utility and can be run from the command line.

The Hopper

The purpose of the Hopper is to interpret CORES packages, determining their next destination. The hopper will receive an incoming CORES package and unwrap just enough information to determine what is the next phase in the process. The possible phases that a node may execute are: send the request to be served by the Proxy Server, make a request to a website on behalf of the originating node, or just move the request across the route

Proxy Client

The purpose of Proxy Client is to convert CORES packages from the Substratum Network into regular (non-clandestine) requests and convert the regular (non-clandestine) responses back into CORES packages for the Substratum Network. A Proxy Client on some distant Node is the component that does the non-clandestine communication with the greater Internet on behalf of the originating node. This is what would be used by an Exit Node. The Proxy Client converts your CORES package back into a regular request, gets the response, and wraps it in a new CORES package. That CORES package goes back onto the Substratum Network to continue on the Route back to the originating node.

Proxy Server

The purpose of Proxy Server acts as the only server your device can get information from. It is a proxy for all your requests. For example, every website you request when running Substratum Node will be delivered via the proxy server. As far as your system is concerned this server represents every server.

Neighborhoods

Each Substratum Node in the Substratum Network contains a subsystem that we call the Neighborhood. The Neighborhood is responsible for keeping track of other Nodes in the network, remembering how (and whether) they're connected to one another, and storing other information about them as well. Currently, for example, the Neighborhood keeps track of whether nodes are running in Bootstrap Mode which function to provide gossip about the network rather than routing data. Whenever a consuming Node wants to send a CORES package through the network, its Neighborhood is responsible for analyzing the network and determining what route the CORES package should take. Neighborhood data is stored within an in-memory database. This creates a level of security that allows a node to shut down and be free of any knowledge it originally contained.

Bootstrap Nodes

When a node first starts up it will contain a limited set of instructions to reach out and contact available Bootstrap Nodes. A bootstrap node provides a collection of available node addresses for this new node joining the network. This allows the new node to get a kick start onto the network. Bootstrap Nodes will not perform the same functions as a typical node. While they will provide other nodes with information to build Neighborhoods, they will not route requests and they will not be monetized.

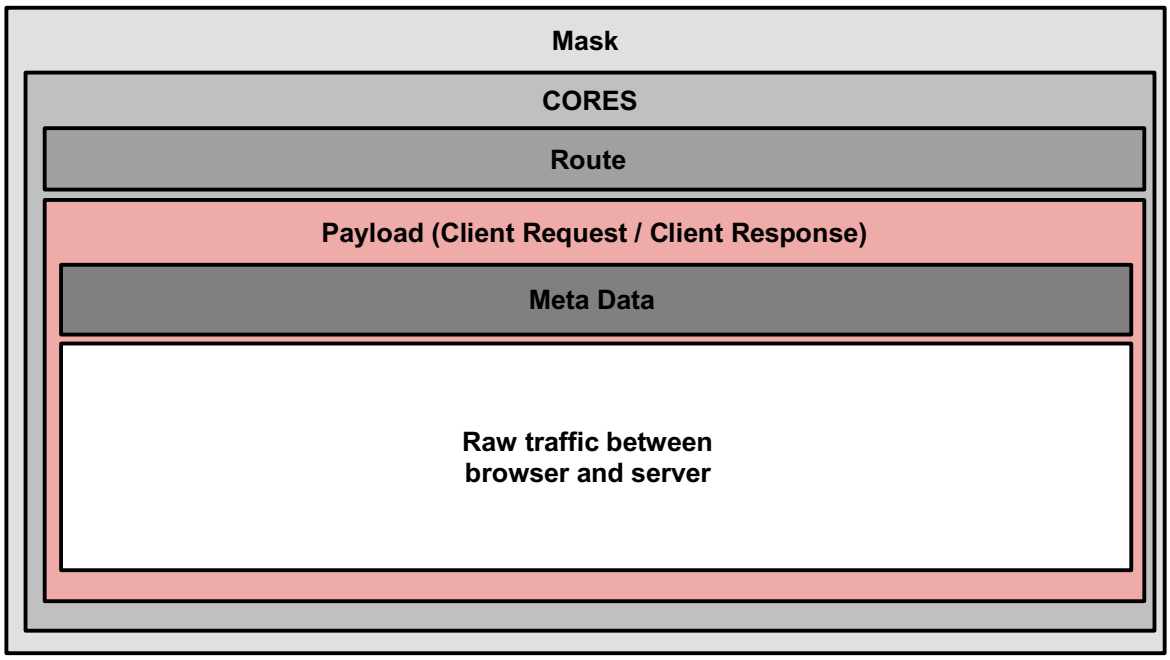
The life span of Bootstrap Nodes may come to an end in the future as more and more nodes join the network. As more nodes come online the use of social connection features will help with bootstrapping nodes. A user of the node can get the address of another node and add it manually to their Neighborhood. This information can be gathered through social means just the same as other open source projects are shared and used socially across the web.

Blockchain Bridge

The job of the blockchain bridge is to communicate with the actual blockchain. The blockchain bridge does not connect directly to the blockchain. It leverages the Proxy Client to make the connection. This allows nodes to subvert restrictions from access to blockchain information in the same way that Node subverts website restrictions via the Proxy Server.

Clandestine Routing / Masking

As the network grows and becomes available there will be a time when outside forces will want to intercept or prevent the network from being utilized in the way it was intended. To create another barrier to prevent malicious intent, packages will be wrapped in a mask. The first mask will be basic encryption methods we see today. These masks will provide an extra layer of security similar to typical HTTPS traffic we see today. As development of more in-depth features for the network continues the goal is to create Clandestine routes. Each CORES package will be wrapped in a layer of obscurity that will resemble different types of traffic. Traffic patterns when viewed over time can be distinguishable. By masking packages in random traffic patterns, it deters typical packet sniffing and traffic analyzation methods from seeing patterns and blocking requests and responses. This entire layer will wrap around CORES packages and its various components before being handed off as seen in the diagram below.

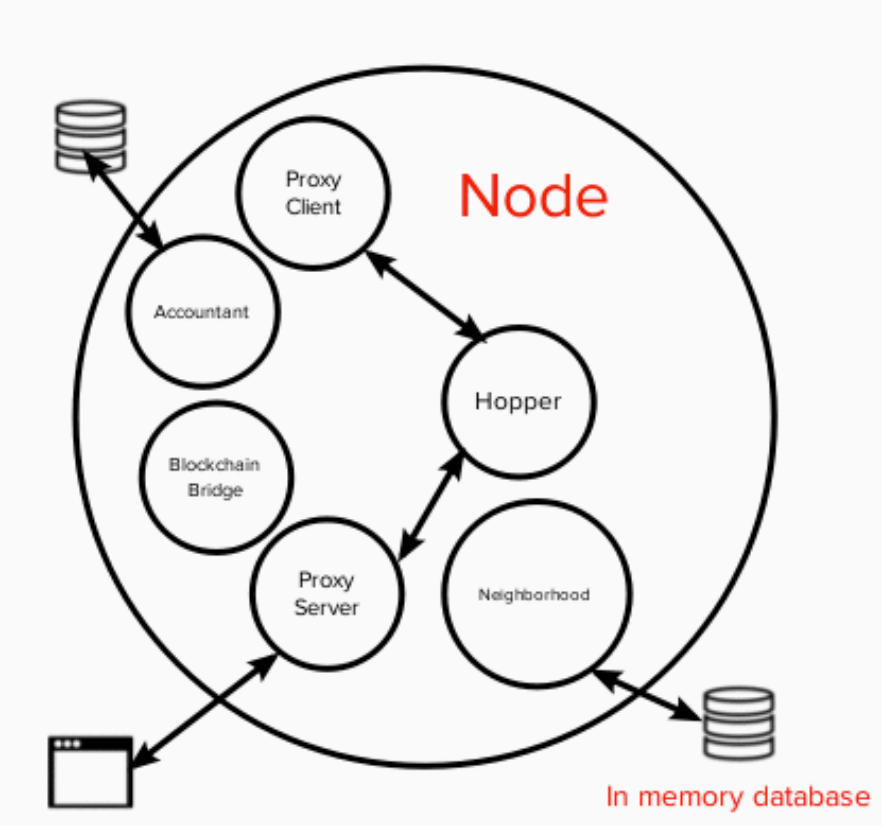


Gossip Protocol

At certain times, the Neighborhood will trigger a round of Gossip. When this happens, the Neighborhood will send a CORES package to each of the Nodes to which your Node has a direct TCP connection. * This CORES package will contain part of what your Node knows about the current state of the network, including what it has learned from other Nodes when they have sent Gossip to it. When you start a decentralized Node, it needs to know how to locate just one other Node in the network; but these Gossip messages are how it strengthens its position over time and becomes an integral participant in the Network.

Since every Node sends Gossip periodically, information about your Node will eventually percolate out across the network, and information about the farthest reaches of the network will eventually reach your Node.

*The proposed Monetization configuration will not occur a charge since each of these Nodes is right next to you. no Node but your own sends any data on your behalf, and therefore there's nobody to pay.



The Substratum Network Step by Step

Now that you have an understanding of each component and its function, let's walk through a step by step guide on a complete end to end request using the Substratum Network. The guide will demonstrate how each node relays information across the wire.

1. Substratum Network starts with a computer running Substratum Node Software
2. Each Node belongs to a Neighborhood
3. A Neighborhood is composed of multiple Substratum Nodes. Each node is only aware of its immediate neighbors, this helps ensure anonymity for the Nodes and boosts security by preventing the level to which any attacker could potentially penetrate the network.
4. When someone requests a web-site the Node creates a request package for one of its immediate neighbors. The contents of the request package consist of the following:
    a. An encrypted route stack for going to and from the destination
    b. An encrypted package for each node in the route stack
    c. An encrypted payload with the original web request
5. The Originator Node creates a predetermined route stack, both to and from the request endpoint, using its Neighborhood.
6. A client, like a web-browser such as Google Chrome, submits a request to a website.
7. Substratum Node creates an encrypted package with encrypted parts for each node along a route path. The route path is predetermined by the originator node and contains different request and response paths.
8. This limits the potential attack surface because the request path to the HOST and the path back to the CLIENT are different.
9. The encrypted package is sent to an IMMEDIATE neighbor. Each Substratum Node only knows the IP addresses of IMMEDIATE neighbors, this also works to limit the potential attack surface.
10. The neighbor receives the package & decrypts it. Only the Node with the matching fingerprint can decrypt the package
11. Here's how a node decrypts the package:
    a. The recipient node decrypts the next set of instructions, not the entire package.
    b. The instructions for the next hop tell the node to relay the request to the next node, by public key, in the route stack.
    c. Instructions for an additional node tell the recipient to relay the request to the next node. (A minimum of three hops allows only compartmentalized knowledge of the network.)
12. Since each node is not aware of IP Addresses outside of its immediate neighbors the ability for attack on the network is greatly reduced.
13. The final hop to the destination includes a payload and instructs an Exit node to create a web request and response on behalf of the originator using the request data provided by the Client (such as Google Chrome or your mobile browser).
14. A response will return along a unique route which only further boosts the security and anonymity of the network.
15. The return path behaves like the request path but includes the encrypted response for the client. Only the originating node is able to decrypt the response & hand it to the browser to be rendered

Constraints

In order for this process to work correctly, safely, and profitably, there are certain common-sense constraints on what a Node should and shouldn't do.

Connection

Perhaps chief among these constraints is that you should keep your Node available at the same public IP address as long as you can. Suppose your Node is listed by another Node's Neighborhood in a route for a CORES package, and by the time the CORES package arrives at your Node, your Node cannot be contacted in the way the originating Node's Neighborhood expected it to be. In that case, the CORES package will fail to route, and it will have to be sent again; but in order to keep itself from making the same mistake, the Node that failed to contact your Node will A) make a note not to route any more data through your Node, and B) Gossip this information ("Don't route data through me to him; I can't contact him") out to other Nodes. In this way, even a brief period of disconnection can seriously affect the amount of (potentially) profitable routing work your Node will be given.

We have plans to make certain public-IP changes survivable, but currently none are. If your public IP address changes, your Node traffic will drop to zero and stay there; you'll need to kill your Node and restart it to get it back on the Network.

Device Reliability

A device such as a laptop that can be suspended or lose battery power may not be ideal for running an effective Node. Although a Node occupies a tiny footprint on most devices in regard to Size, CPU, and Memory Consumption. Living on a device that suspends, shuts down, or has and unreliable power supply is not recommended. The Node should stay running and connected as long and steadily as possible. Future features may include keeping track of statistics such as how fast a node has proven to be and whether they're acting suspiciously.

Warm-Up Delay / Three Hop Minimum*

Each Node in the network will know the IP addresses of the few Nodes that are directly connected to it, but it will not know the IP addresses of any other Node in the network. There are a few security-related implications of this.

To be effective an Originating Node requires a route at least three hops long to make sure that none of the Nodes on the route knows all the IP addresses in it. Therefore, a route less than three hops long is intrinsically insecure.

An Exit Node for a two-way route would need to know the public key of the originating Node, so that it can encrypt the response payload. Therefore, the exit Node should not also know the IP address of the originating Node; otherwise it would be able to associate request URLs (TLS) or entire request/response transactions (HTTP) with originating IP addresses.

For a Neighborhood to produce valid routes it needs enough Nodes that it can produce a minimum three-hop route, but not so many connections between Nodes that it can't find an exit Node that isn't directly connected.

All this means that it may take a significant interval of time, marked by the periodic arrival and departure of Gossip, before a newly-started Neighborhood is complete enough to be able to generate routes to allow its owner to originate traffic. A Neighborhood will be ready to route other people's data as soon as it has two immediate neighbors that are not Bootstrap Nodes. Of course, in order for a Node to route traffic it must be built into a route by another Node, and it may take some time for another Node to assimilate and choose it as well.

* The current version of Substratum Node is not yet secure, it has a default route-length minimum of two hops rather than three, because it makes testing easier while still presenting all the conditions we need to test. Upon release of the first production version of Substratum Node the default route-length will change to three hops as mentioned above.

Best Performance Measures

The best computer on which to run a Substratum Node is one whose public IP *never* changes: one whose ISP has granted a static IP address. Failing that, the best way to run a Node is on a non-portable computer that is using a wired Ethernet connection. (Sending data through a wire keeps it far more secure than broadcasting it through the air.) Failing *that,* the best way would be on a WiFi-connected computer that may move around in the territory of a particular router but doesn't leave it and is never suspended or hibernated.

Monetization

Substratum plans to incentivize users running nodes by rewarding them in cryptocurrency. This digital form of monetization allows the system to remain decentralized and have value across the globe. To break down how the system will self-govern and distribute this currency we will have to identify key features and protocols that are necessary for successful monetization.

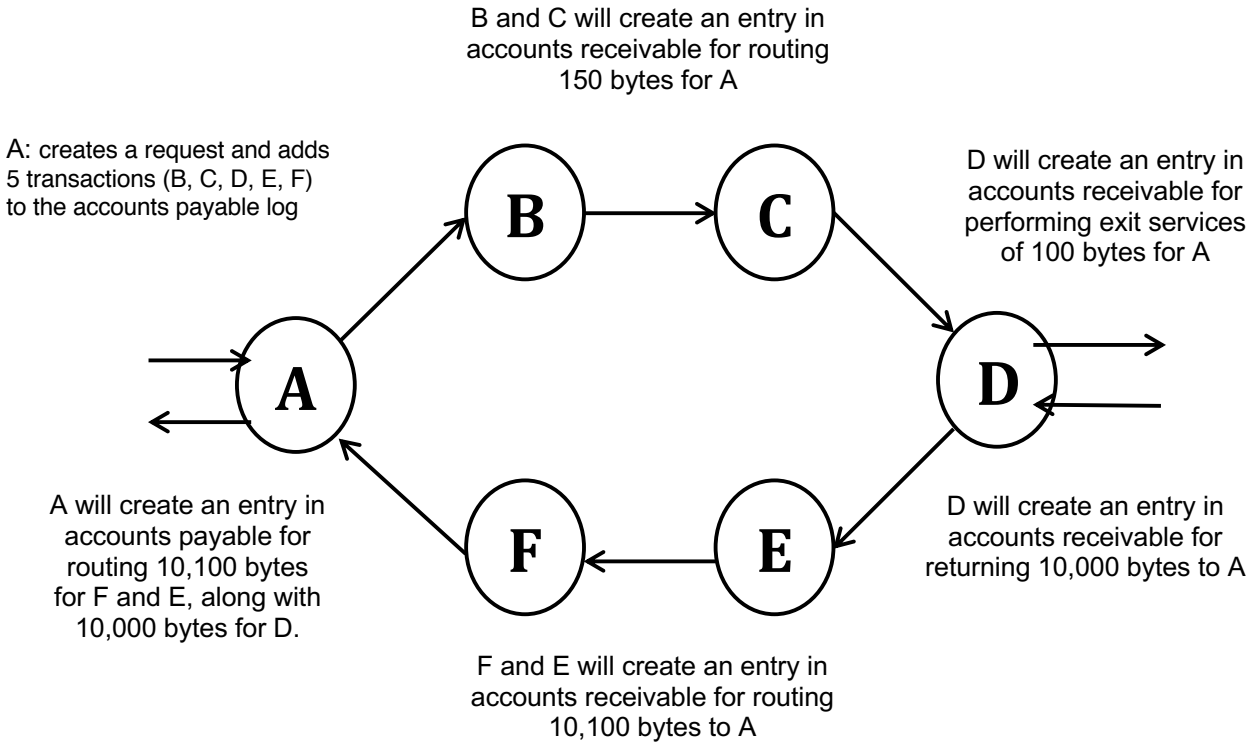
What we are counting

The first problem we had to solve when understanding how we can monetize the network is what to count. Multiple iterations of network traffic were analyzed to see how data could be counted and logged without compromising security. These iterations resulted in the counting of bytes transferred within CORES packages. The platform also counts how those bytes are used within the steps contained within CORES packages. The network accounting will monitor routing a package along with providing exit services as well. This is to create higher rewards for exit nodes.

1. **Bytes**
2. **Routing Services**
3. **Exit Services**

How is it counted

To avoid the concept of a central authority, nodes must be provided with a mechanism for storing what would be considered accounts payable and accounts receivable data. Each node keeps track of what it owes other nodes for requests. It also keeps track of what is owed for transmitting routes from requests for other nodes. This creates a self-governed policy that provides several mechanisms for validating both good and bad actors. More importantly it provides a method to adjust pricing on a per node basis based on factors that the node owner would want to make adjustments for. This is discussed more in the how it is rewarded section.



How is it rewarded

The best way to describe how the value model is applied to this form of logging transactions is a free market approach. By allowing nodes to set their own price a central system does not need to monitor costs, and payout across a massive network. The goal is to allow nodes to set prices for their services based on a number of factors. Some of the factors include regional availability, uptime, upload and download speeds, reliability, truthfulness in transactions, and possibly availability in censorship oppressed regions. In the same way that node owners can name their price, node consumers will want to identify thresholds in their spending. An example how the node network plans to achieve this is to create configurations that allow consumers to set spending limits per transactions. Node users should be able to change their spending habits based on the quality of nodes by paying higher costs for premium nodes or take a more budget friendly approach to node use.

What the future holds

This is a living document that will be updated as the product continues to evolve. Below are features that are important for how the network will function but are still in the analysis and research phase of development. As the work becomes clearer and more tested these features will be documented and updated within their respective sections of this paper. Here are some details to look forward to.

- **Validating monetization transactions through consensus**
- **Handling bad actors within monetization**
    a. **Actors who don't pay**
    b. **Actors who overcharge (cook the books)**
    c. **Route failures**
- **Exit Services and banning**