“ORB”
DECENTRALIZED FUNCTION AS A SERVICE (DFAAS)

Project ID – 18-070

Project Proposal Report

R.G.D Nayomal
S.K.N.U Tissera
T.H.A.K Silva
A.T.Nimansa

Bachelor of Science Special (Honors) Degree in Information Technology

Department of Software Engineering

Sri Lanka Institute of Information Technology
Sri Lanka

March 2018

“ORB”

DECENTRALIZED FUNCTION AS A SERVICE (DFAAS)

Project ID - 18-070

Project Proposal Report

(Proposal documentation submitted in partial fulfilment of the reqirements for the Degree of Bachelor of Science Special (honors) in Information Technology)

Bachelor of Science (Special honors) Degree in Information Technology

Department of Software Engineering

Sri Lanka Institute of Information Technology

Sri Lanka

March 2018

Declaration

We declare that this is our own work and this proposal does not incorporate without acknowledgement any material previously submitted for a degree or diploma in any other university or Institute of higher learning and to the best of our knowledge and belief it does not contain any material previously published or written by another person except where the acknowledgement is made in the text.

Group Members

Name	**Student ID**	Signature
R.G.D Nayomal	IT15027948	
S.K.N.U Tissera	IT15120212	
T.H.A.K Silva	IT15136916	
A.T. Nimansa	IT15419910	

The above candidates are carrying out research for the undergraduate Dissertation under my supervision.

Signature of the supervisor: …… Date: ……

Abstract

The purpose of this research is to implement an autonomous, user oriented, decentralized Function as a Service(FaaS) provider that can replace centralized, single-authority FaaS provider which can address prevailing drawbacks of global cloud infrastructure related to serverless architecture. Serverless architectures refer to applications that significantly depend on third-party services (known as Backend as a Service or "BaaS") or on custom code that's run in ephemeral containers (Function as a Service or "FaaS"), the best-known vendor host of which currently is AWS Lambda. The existing serverless architecture suffers from drawbacks such as Vendor control, multi tenancy problems, vendor lock-in, security concerns, lack of debugging and monitoring tools, difficulty of managing of granular functions and architectural complexity. The proposed system resolves the vendor control, multi tenancy, vendor lock-in, security concerns and managing of granular functions by decentralizing the granular functions throughout a peer to peer network in order to provide a decentralized FaaS. A granular function refers to an atomic method/function with a single responsibility (such as addition, image recognition, IoT connector) hosted in the network. The "Orb" is a peer to peer network consisting of peers(nodes) and super peers (supernodes / trackers). The node discovery of the network is done using the principles of "Satoshi Client Node Discovery". The user could sign up to the network as a node or a supernode by installing client or the server software provided depending upon the requirement. Node (represents a personal computer) is a peer, providing functionality of implementing, deploying, hosting functions. Supernodes keeps a track of peers, live supernodes, hosted granular functions etc. The user could deploy an atomic granular function to the network through the client software. For deployment, user will be charged in cryptocurrency(Ether). The payments model of "Orb" depends on ether smart contracts which uses the blockchain[17] technology. The deployed function will be sent to the super nodes and distributed across a set of peers throughout the network to achieve high availability, reliability, integrity. The confidentiality of a deployed function is achieved through techniques such as public key private key encryption, proper packaging and containerization. The users can limit the access to the function by keeping it as a private and opening it to a selected set of users. For hosting a function, the user will be paid in "ether" considering the no of requests served and uptime. The methodology of testing is to implement a system which runs on a local network behind a Network address translator. By measuring the response time by calling the function against active number of peers will provide the analytics data about the reliability and availability of the system. In future, "Orb" will support decentralized APIs.

TABLE OF CONTENT

TABLE OF FIGURES

1.0 Introduction

1.1 Background

The proposed decentralized Function as a Service solution is a unique research area related to data communication which changes the definition of serverless computing. Currently most systems are centralized and has a single-authority. The data is owned by, around 150 large scale companies. These companies will decide how the data is manipulated, charged and controlled. The proposed system will benefit the users by giving control of their and power of information back to them.

Serverless architectures refer to applications that depend on 3rd party services (knows as Backend as a Service or "BaaS") or on custom code that's run in ephemeral containers (Function as a Service or "FaaS"), the best-known vendor host of which currently is AWS (Amazon Web Services) Lambda (FaaS provider). The name "serverless computing" is used as the end user doesn't have to code the backend [15] to run. Serverless code can be used together with traditional architectures, such as microservices [16]. For example, part of a web application could be written as microservices and another part could be written as serverless code. Alternatively, an application could be completely serverless if it's written only using granular functions without the backend [15]. FaaS helps developers to code the functions without paying attention to the server infrastructure. But serverless architecture has a notable set of drawbacks since server infrastructure is maintained by the vendor. Vendor control, multi tenancy problems, vendor lock-in, security concerns, lack of debugging and monitoring tools, difficulty of managing of granular functions and architectural complexity as some of the drawbacks. The proposed system resolves the vendor control, multi tenancy, vendor lock-in, security concerns and managing of granular functions by decentralizing the granular functions throughout a peer to peer network in order to provide a decentralized FaaS. A granular function refers to an atomic

method/function with a single responsibility (such as addition, image recognition, IoT connector) hosted in the network. “Orb” has 5 major challenges.

1.1.1 Decentralized Service Deployment
1.1.2 Network Infrastructure
1.1.3 Function as a service model
1.1.4 Payments model
1.1.5 Security

An introduction to the major challenges is given below.

1.1.1 Decentralized service deployment

After developing a certain service (API, Lambda function), the user needs to host it in order to use it. Since complex apps might consume higher computational power and cost might be high, the developers or the owners of the app might tend to host it on centralized cloud computing platforms like Amazon Web Services. There can be outages [11] since all centralized systems come with a single point of failure [12]. The most recent outage being the “AWS S3” outage [1] where 1000s of websites suffered a downtime. The proposed solution can be advantageous since the single point of failure [12] is eliminated. The existing cloud computing platforms such as AWS EC2 and GCP (Google Cloud Platform) Compute engine, the user is charged for the storage, memory and the number of cores of the CPU and more importantly, the code that’s deployed to the above-mentioned providers will be owned by them. Decentralizing functions will result in the elimination of the single authority and reduce the costs.

1.1.2 Network infrastructure

The torrent technology has existed from 2001 to present. It is not fully P2P. Centralized servers called trackers are used to keep a track of the nodes and the files. When a torrent client requests a file through trackers, the relevant set of peers will be given and the download starts. When this does, the user can enable DHT (Distributed Hash Table) which will keep a track of live peers. Now the torrent can request a file fragment from peers and continue the download even without the tracker server solely dependent on the peers. The downside of torrents is it' needs a centralized server if it uses a tracker to identify peers. If the torrent is dependent only on DHTs to find peers, at first the client must know about a live peer to start and then the DHTs will update and the network will grow.

This would be the ideal technology for the proposed infrastructure. The proposed system will not split files as in torrents but the distribution of the functions must be handled similarly to the BitTorrent protocol. However, the challenge is to implement the BitTorrent protocol in a way to suite the "Orb". A network of tracker (super nodes) servers will help the users to locate the deployed functions and monitor the usage by acting as a gateway.

1.1.3 Function as a Service

Function as a service (FaaS) is a category of cloud computing services that provide a platform allowing customers to develop, run, and manage application functionalities without the complexity of building and maintaining the infrastructure typically associated with developing and launching an app [2]. Building an application following this model is one way of achieving a "serverless" architecture [4], and is typically used when building microservices [3] applications.

Serverless doesn't mean that "there are no servers" but the allocation of server resources will be automatically managed and handled by the service provider.
In the proposed system, the user doesn't have to configure server resources such as storage, memory. The user deploys a function and an endpoint will be generated and sent almost immediately. The basic infrastructure for hosting is a p2p network[23].

Even though configuration overhead is eliminated. The functions cannot be edited or removed easily due to the decentralized nature of the system.

1.1.4 Getting paid with Ethereum through blockchains

Ethereum is an implementation of blockchain[18] technology. It allows users to create executable programs called "Smart Contracts"[6]. Using these smart contracts, the users of the network can perform certain actions. For example, they can perform voting, distributing funds, crowdfunding, fundraising etc.

1.1.5 Security

Decentralized nature of the system makes orb resistant to DDoS (Distributed Denial of Service) attacks. Usage of supernodes to route requests when a client is consuming a hosted function makes the system more robust and resistant to man in the middle attacks since the route to the consumed function will only be decided by the supernodes. Most of the financial models that exist today are built upon trust. But Orb brings an unprecedented level of trustlessness for the users by the elimination of a third party when it comes to payment. The users will initially subscribe to an Ethereum contract where the need for trust is eliminated.

When it comes to privacy, the functions will be anonymous in the system. A function will only have a unique hash to access it. Only the user who deploys this knows about the ownership of the function. To make the hosted functions tamper-proof, an executable will be deployed (no source code). Whether the functions are developed as executables, still they could be reversed engineered. To prevent this, Orb uses a hashing system where the function is initially run through a hash and that hash will be distributed through the supernodes. When a node connects to a supernode initially, all the hosted functions will be validated before exposing the function for consumption.

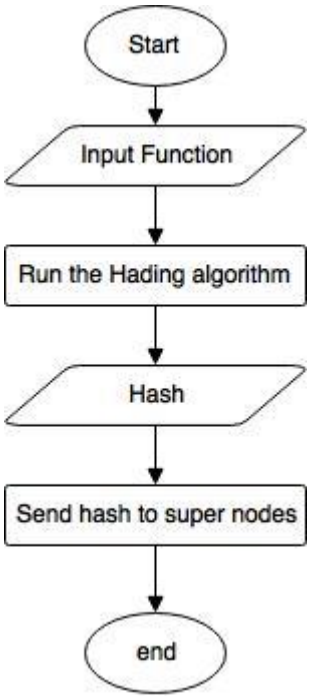


Figure 1: Generating a unique hash before deploying a function

1.2 Literature survey

1.2.1 Interested Systems

1.2.1.1 AWS Lambda[8]

Amazon web services lambda is also providing server space for the functions to be deployed. With AWS Lambda[8], we can run code for virtually any type of application or backend service - all with zero administration. AWS Lambda runs the code on a high-availability computer infrastructure and performs all of the administration of the computer resources, including server and operating system maintenance, capacity provisioning and automatic scaling, code monitoring and logging.

1.2.1.2 Steemit

Steemit[9] is a social network and content rewards platform that makes the crowd the beneficiaries of the attention economy. It does this be rewarding users with STEEM. Steemit is a social media platform that works by having the crowd reward the crowd for their content. It does this thanks to the Steem blockchain[18] and cryptocurrency; Steem is 'minted' daily and distributed to content producers according to the votes they get. Most social media sites extract value from their user base for the benefit of shareholders alone. Steemit is different, it's a new kind of attention economy. By connecting with the Steem blockchain (which is decentralized and controlled by the crowd), Steemit users receive all the benefits and rewards for their attention.

1.2.1.3 IPFS

IPFS[13] is a distributed file system that seeks to connect all computing devices with the same system of files. IPFS[13] is a versioned file system that can take files and manage them and store them somewhere and then tracks versions over time. IPFS[13] also accounts for how those files move across the network so it is also a distributed file system. This file system layer offers very interesting properties such as:

I. Websites that are completely distributed
II. Websites that have no origin server

1.2.1.3.1 Content Addressing

Instead of referring to objects (pictures, articles, videos) by which server they are stored on, IPFS refers to everything by the hash on the file. The idea is that if in your browser you want to access a particular page then IPFS will ask the entire network "does anyone have this file that corresponds to this hash?" and a node on IPFS that does can return the file allowing you to access it. IPFS uses content addressing at the HTTP layer.
IPFS Objects

IPFS is essentially a P2P system for retrieving and sharing IPFS objects. An IPFS object is a data structure with two fields:

I. Data - a BLOB of unstructured binary data of size < 256 kB.
II. Links - an array of Link structures. These are links to other IPFS objects.

A Link structure has three data fields:

I. Name - the name of the Link.
II. Hash - the hash of the linked IPFS object.
III. Size - the cumulative size of the linked IPFS object, including following its links.

1.2.1.4 Storj

Storj is a platform, cryptocurrency, and suite of decentralized applications that allows you to store data in a secure and decentralized manner. Storj can be faster, cheaper, and more secure than traditional cloud storage platforms. Faster because multiple machines are serving you your file simultaneously, cheaper because you are renting people's spare hard-drive space instead of paying for a purpose-built data center, and more secure because your file is both encrypted and shredded. Storj uses blockchain features like a transaction ledger, public/private key encryption, and cryptographic hash functions for security. The decentralized aspect of Storj means there are no central servers to be compromised, and with client-side encryption, you are in control of the keys to your files.

1.3 Research Gap

Decentralized FaaS is a new concept which didn't exist up to now. Despite the FaaS Service providers which exist today, none of them offers a decentralized solution.

Although there are many service providers which provide server space to deploy our application, most of them work in a centralized manner. But there are no solutions which provide this service in a decentralized manner. As for the FaaS is concerned, there's no proper way or existing systems which use the concept of decentralized FaaS as of today.

The reason why someone would decentralize a function is the avoidance of any central points of failure. Developers can create, deploy and monetize serverless functions completely anonymously and users can consume them in full privacy. In a typical environment, the traffic can be intercepted through a single point using a load balancer and routed to different servers. But in "Orb", each supernode acts as a load balancer itself.

Function calls are mediated by super nodes depending on the latency and number of requests per second automatically balancing the traffic.

The attractive feature why a user would host a function on their own machine is, hosting a function in their own machine will make the node eligible for payment through an Ethereum contract, a user subscribes to, at the time of deployment. The payments model works as pay per request terms. More traffic will generate more income. The advantage of such a model is that user itself can host and deploy at the same time. Then the deployment cost can be prevented by hosting and contributing more towards the growth of the network.

The user will use Orb instead of uploading it in AWS or any other because, decentralizing cuts the cost, makes the owner completely anonymous, all the transactions made are verified and more contributions from the user to the network will be paid in terms of ether. Data ownership and central authority are prevented completely from this approach. At the end the user gets rewarded which will be proportional to the contributions made to the system.

Decentralization comes with drawbacks. Once deployed, how can the owner modify or deprecates the function? In a decentralized environment, it's difficult to update functions. Updates should be managed as versions of that function and tracker always choose the latest version of the requested function. The older version will get deprecated once the user issues a request.

What are the measures for Security at nodes in the system? By default, the system will not have a central point of failure, which makes the system fully tolerant to DDoS(Distributed Denial of Service) attacks. The system is also resistant to Man in the middle attacks as function calls will be redirected through supernodes. Each time, the user might call 2 different nodes in 2 different locations by avoiding a specific route. Only an executable of

the function will be deployed (without the source code) as a security mechanism. To ensure that the decentralized functions are not changed, the supernodes will automatically verify by comparing a dynamically generated hash and the existing hash, each time a peer comes live.

1.3.1 Innovation

The area of Decentralized applications has gained traction recently. With static content sharing networks such as IPFS, and browsers such as beaker browser, decentralization of the internet has begun. Introduction of Orb with its unique capability of providing serverless functions replicated and decentralized through a network of nodes and a unique rewarding system using smart contracts, the users can now deploy serverless functions which could be used as servers in a more decentralized internet. The commercialization of the orb is achieved by attracting users to use the network as service providers or service users. The market tractions could be gained as the Orb is the first to market.

1.4 Research Problems

1.4.1 Centralized Systems

1.4.1.1 Single Authority

AWS provides EC2(Amazon Elastic Compute Cloud) helps to create virtual machines which help to run their own computer applications. The users can easily deploy web services/applications. EC2 uses technologies like auto-scaling and have higher reliability (availability rating of 99.9974 percent, CloudHarmony reported). Hence more users are attracted to AWS and from an outside perspective, AWS' dominant share (40 percent) of

the public cloud services seems like a monopoly in nature. Shares owned by Microsoft Azure (15 percent), Google's cloud (7 percent), and the remaining small percentages picked up by IBM, Oracle, and other small players in cloud service compared to AWS. Therefore, if AWS suffers from a downtime, users might feel as the entire internet is down, similar to the massive internet outage happened on February 28th, 2017

1.4.1.2 Your data belongs to a 3rd party organization.

Deploying a function to a third-party services provider always has its downsides. The organization automatically gets the ownership and access to the data. Although service providers like AWS can be trusted, there is always a risk.

1.4.1.3. Most of the systems contain a central point of failure

The classic definition of single point of failure[12] is of a potential risk posed by a flaw in the design, implementation or configuration of a circuit or system in which one fault or malfunction causes an entire system to stop operating. In a small or medium-sized business which runs on a single server, this problem is there. Since that business might run low on profit affording for cloud computing / AWS will hardly be possible.

1.4.1.4. Most of the systems are not scalable by default

Systems and functions within will get updated continuously and processes might get complex hence the computational power for them might increase. Most of the systems lack auto-scaling or manual scaling ability due to many reasons. If the business depends on a single server and if they need more space, they have to vertically scale up the server. But the hosted system will suffer a downtime.

1.4.1.5. Centralized systems tend to generate more traffic

Amazon Alexa is an Amazon's cloud-based voice service available on tens of millions of devices from Amazon and third-party device manufacturers. Using a function implemented in AWS Lambda[8] we can connect many Alexas to the AWS cloud to an external API and call it. When many devices are connected, will generate a lot of traffic and lambda will get scaled automatically balancing the load. Although it will reduce the traffic through auto-scaling the process will cost extra money.

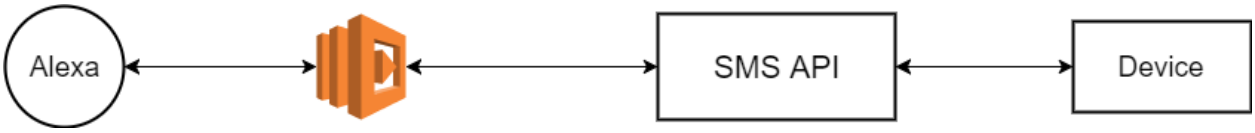


Figure 2 - Alexa, AWS Lambda integration

1.4.1.6. The security of the data is a concern.

Even though there are client side and server-side protection. The organizations will have the full control over your data.

1.4.2 Existing Serverless architecture

1.4.2.1 The function is accessed through 3rd party vendor.

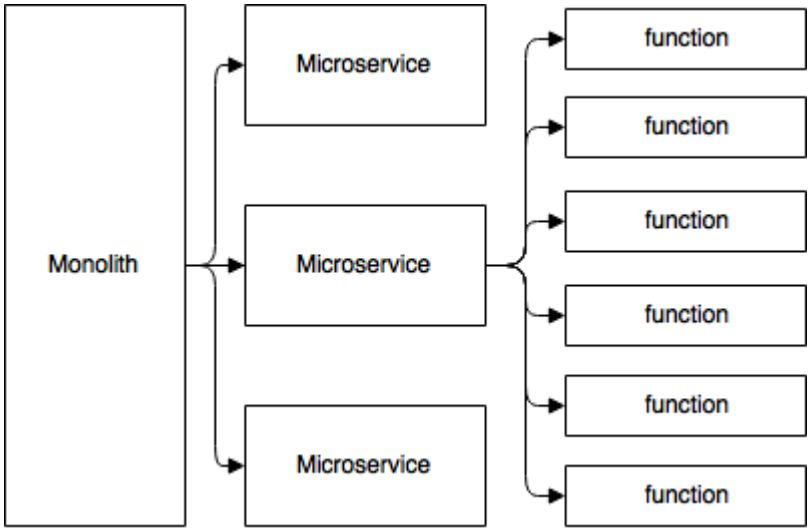
The problems such as vendor control, multi-tenancy problems, vendor lock-in comes in this category. If the vendor updates its APIs which provides access to the cloud functions. The hosted functions will suffer downtime.

1.4.2.2 Lack of debugging and monitoring tools

To debug the functions, the developers have to use vendor-specific tools.

1.4.2.3 The difficulty of managing of granular functions and architectural complexity as some of the drawbacks.

Figure 3 - Breakdown from Monolith to serverless architecture



As shown in *Figure 3.2*, breaking down a large monolithic application into a set of functions is complex. The developer must decide the interactions before implementing. This is an overhead.

1.4.3 Underutilization of computational power

A 1985 Cray-2 supercomputer had a processing power of 1.9 GFLOP and a processor of 244Mhz but Apple iPhone 4 has an 800Mhz processor with a compute performance of 1.6 GFLOP. But most of the time, these devices are kept idle, without utilizing the CPU power, the need for computing resources is growing at a fast pace. IoT based systems,

machine learning and deep learning, complex algorithms and other sophisticated solutions being deployed in every domain and industry are raising the demand for stronger cloud servers and more bandwidth to address the minute needs of enterprises and businesses. By decentralizing the applications will provide a platform that enables participants to lend and borrow computing resources while generating income.

2.0 Objectives

2.1 Data communication between the nodes

2.1.1 Finding an efficient mechanism to explore the possible ways of bi-directional communication between private and public networks to accomplish the following

- To decrease latency of data communication between the nodes.
- To improve communication between nodes in the same network efficient
- Determining secure ways to store functions in nodes

It's possible to find peers in a local network since all the nodes are in the same network but transmitting data and finding peers between public and private networks is a challenge. To achieve this, the proposed way is to use a supernode in between where all the communication is done through the supernode. Implementing the underline server and client software should be done to manipulate requests through the network.

2.1.2 Determining secure ways to store functions in nodes finding out a mechanism to revert / edit a function deployed in the network

2.1.3 Finding out the possibility of creating a new protocol to transmit data between the private and public networks

2.1.4 Finding out a mechanism to revert / edit a function deployed in the network.

Due to the decentralized nature of the network, changing or removing a deployed function is a challenge. In Order to overcome this challenge, it's proposed to generate a hash for a specific function which should be unique throughout the network. When a user edits a function, the update is sent to nodes containing the function referenced by the hash. This function needs to be on the client nodes.

2.2 Deployment of Functions to the nodes

Functions should be secure, smaller in size, packaged, and verified to deploy into “Orb”. Creating a deployment framework to deploy functions and their configuration in a way to suite the decentralized network security.

- Packaging
- Configuration
- Obfuscating functions

2.2.1 Finding out a mechanism to determine a common access point (an endpoint) to call functions deployed in to the network.

When replicas of the same function is distributed over the decentralized orb network, there should be a way to consume the functions. But having different routes to consume a function is a problem. Therefore it’s critical to generate a common resource locator to identify the function irrespective of the deployed location. The requests should be routed in the optimum way to reach the nodes and the responses should reach the destinations accordingly. To achieve this, it’s proposed to use decentralized repositories in the super nodes.

2.2.2 Finding out a mechanism to route database a scenario where a single node goes done

2.2.3 Method to retrieve a list of functions to be hosted from super nodes

2.3 Scaling the Orb on internet

2.3.1 Finding out a proper mechanism to boot up and scale the decentralized network.

The initialization of the decentralized network is critical for the Orb to scale. To initialize, there should be a known set of super nodes. These super nodes should be able to broadcast the connected peers to the connecting peers.

2.3.2 Decentralized DNS resolution

2.3.3 Finding out the possibility of using super nodes as DNS servers itself

Addition of extra DNS servers to the orb decreases the network efficiency. Therefore, it's proposed to do the DNS resolution in decentralized manner. But for this to be possible, the super nodes must support the DNS resolution and the required application layer must be implemented separately.

2.3.4 Finding an ideal mechanism to cache or persist data in super nodes

Caching requests increase the performance and reduces the response time. It's critical for the super nodes and nodes to have a mechanism to cache requests.

2.3.5 Finding a mechanism to persist data in client application

2.3.6 Finding out a mechanism to replicate the same function in multiple nodes but the replicated function should act as a single function

Unlike torrents and storej, the functions are replicated over the network, the clients and servers must be able to handle the deployment of the functions accurately.

2.4 Rewarding system

Rewarding users in the system is the key to retain users and increment the contributions which ultimately helps the Orb to grow. Due to the decentralized nature of the system, it's difficult to tamper with data and difficult to implement a centralized payment method. Therefore it's proposed to use smart contracts to reward the users

2.4.1 Payment Model based on requests served by a node, amount of data transferred Via smart contracts

2.4.2 Finding out the possibility of paying the nodes which hosts functions based on served requests

2.4.3 Finding out a mechanism to validate the requests served by a mechanism between the origin of the request and the requests served

2.4.4 Finding out a secure mechanism to prevent DoS attacks which lets a node earns money by sending pings.

2.4.5 Finding a method to determine the value of a smart contract.

2.4.6 Deploying a smart contract to the network with a start price.

3.0 Methodology

3.1 Introduction to Orb

Orb is a decentralized function as a service provider. It consists of 3 major components.

I. Nodes
II. Super Nodes [Trackers]
III. Payments Manager

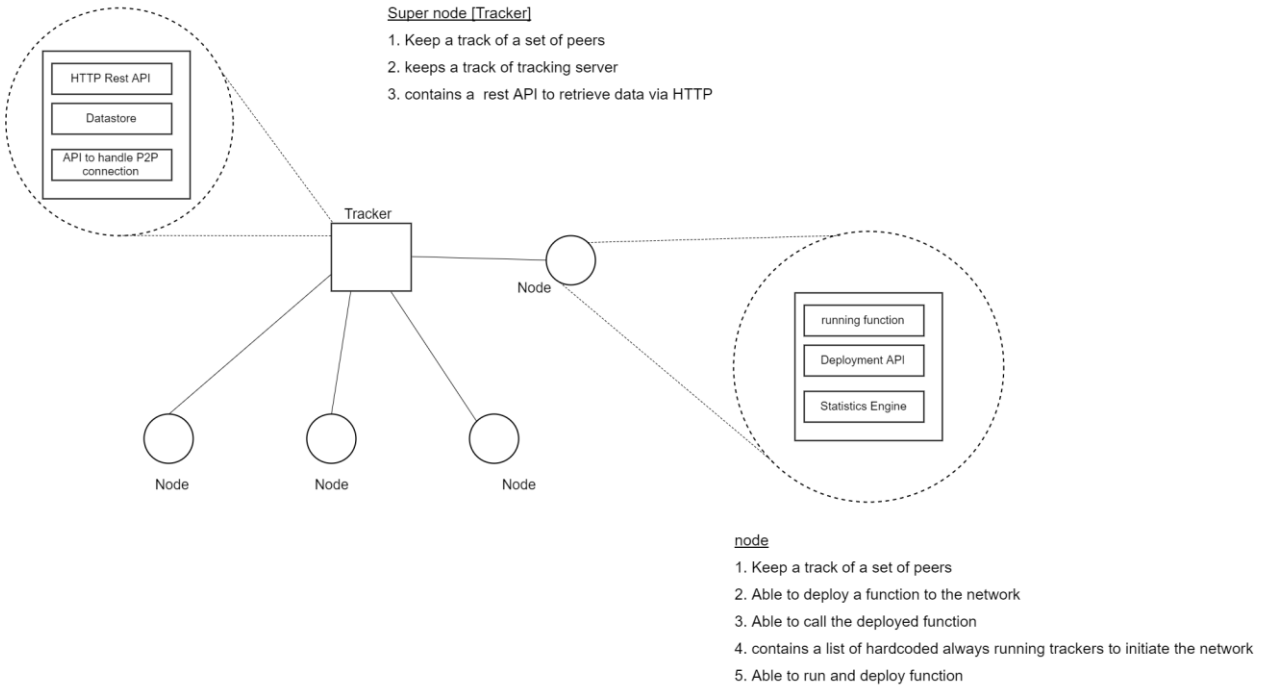


Figure 4 - High-level Architecture

3.1.1 Nodes

A node represents a personal computing device with the Orb client software installed. The responsibilities of the node are as follows

I. Deploying a function to the network.
II. Keeping a track of peers and supernodes - Using DHT and supernodes servers
III. Provide an URL to call a deployed function by the user
IV. Able to run an available function - By running an available function on the client machine will generate income in terms of ethereum for the user.
V. Subscribe to a payments model when deploying an application
VI. A wallet to secure the payments received

The node will consist of the following proposed mechanisms to ensure that the functions deployed are available.

I. Multiple replicas of the same function will be deployed in different nodes
II. A tracker will automatically deploy the particular function when there are no active peers available.
III. Only an executable of the deployed function will be deployed (without the source code) as a security mechanism. To ensure that the decentralized functions are not changed, the supernodes will automatically verify by comparing a dynamically generated hash and the existing hash, each time a peer comes live.

To ensure reliability and to avoid a single point of failure, there will be a chain of supernodes connected to each other.

3.1.2 Super Nodes [Trackers]

Supernode represents a node with the Orb server installed. The responsibilities of the supernode are as follows

I. Keeping a track of peers and supernodes
II. API endpoint generator which generates global endpoints for deployed functions
III. REST (Representational State Transfer) API to communicate with the outside world for data analytics

A user can choose to be a supernode. But in order to be a supernode, it must be on a public network with a domain name assigned to it.

3.1.3 Payments Manager

3.1.3.1 Owner

In the proposed system, the initial deployment will cost the user in terms of ether[5]. A particular user can deploy a function. The decentralized network will decide the deployment process. Also, the network will monitor the usage of the function. If that function exceeds particular request limit the owner of the function has to pay all the hosted nodes through an ethereum contract. When the user puts the requested amount of ether to the contract, will be distributed among the nodes based on the involvement of nodes.

3.1.3.2 Nodes

The nodes will get paid in terms of ether according to the contribution to the network.

3.2 General Functionality

3.2.1 Deployment and distribution of functions in the network

When the client starts the Orb client software, it will send a request to the supernode and the supernode will start to keep a track of the particular node.

When the user uploads a function through the client, the client will ask the user to subscribe to a payments plan (an ethereum smart contract). Then function will run on the same client machine first. Then function source code will be run through a hash function and hashed. This will generate an initial hash which can be used to identify the function. Also, a key pair will be generated for a particular function. Then the function will be converted to an executable file. Then function will be sent to the supernode.

The supernode will initially run the function, generates an URL and send to the deployed user. Then supernode will look for the live nodes list and sends a notification about the availability of the function. The nodes can now choose to run the function or not. This way, the functions will be distributed throughout the network.

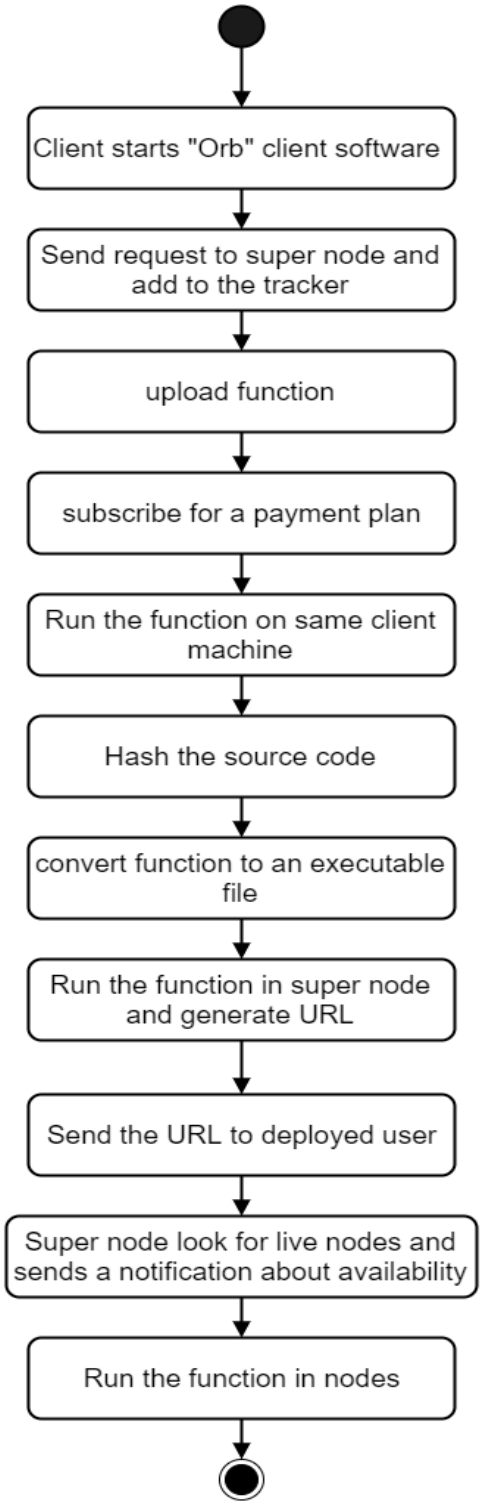


Figure 5 - General Functionality

3.2.2 Updating and deprecation of the function.

The deployed user can initiate a function deprecation or an update. Then this request will be propagated through the network. The updating and deprecation process will be slow since the system is decentralized.

3.3 Sample Scenario

3.3.1 Global Temperature warning system using IoT

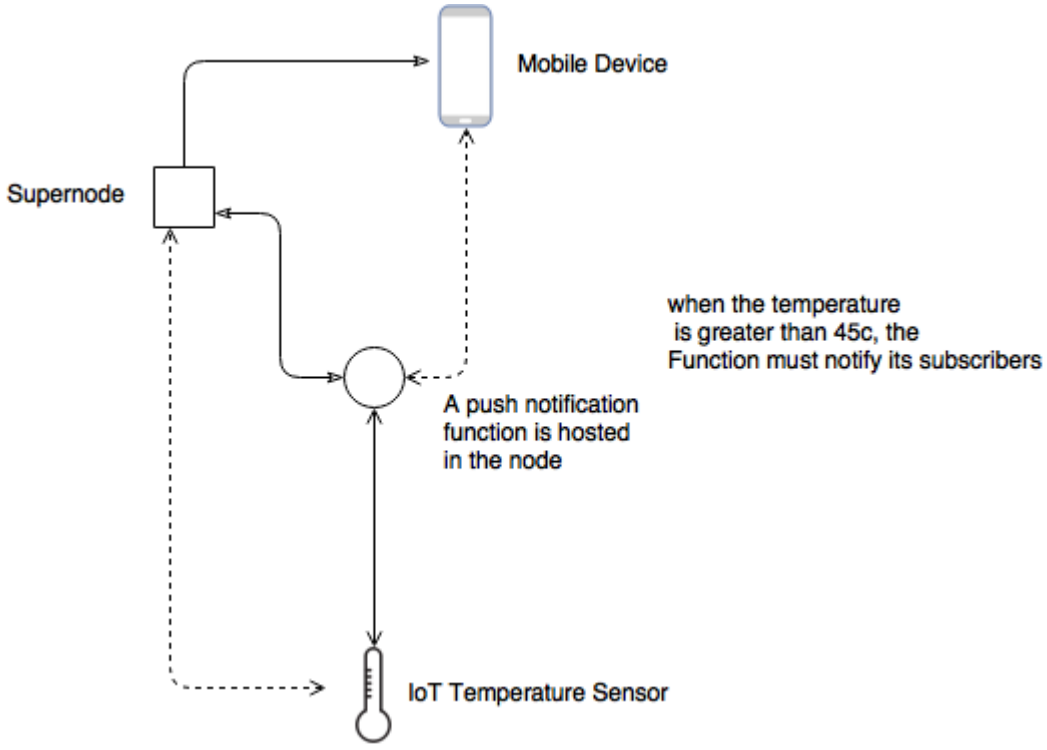


Figure 6- How Orb helps to host a temperature function

Imagine that there are 1000s of supernodes and 10000s of clients contributing to the Orb currently. If a node comes online. It will first subscribe to the nearest supernode and

creates a secure connection right after it's deployed. The node will receive a list of other super nodes and nodes it can connect to. This way, a single node creates a peer to peer connections.

When a user deploys a function, using the above-mentioned process the function will be copied and starts to run on the different nodes. According to above diagram, a push notification function is running on different nodes and there are mobile clients subscribed to the push notification function and consumes the push notification function. The IoT temperature sensor initially connects to a supernode, get a list of peers and establish p2p connections with the other nodes running the push notification function. Now, the IoT function will send real-time data at 5-minute intervals to the nodes, and the function will determine whether to send notifications to subscribers or not. When there's a temperature spike, the IoT sensor will call the function of the peer with the lowest latency. Then the function will get executed and all the subscriber mobile devices are notified.

Since the push notification functions are decentralized and running on 10000s of nodes, there will be a limited traffic. Since the function runs on many nodes, the availability will be very high even if the nodes of offline. The system will tolerate any number of IoT devices since the supernodes decide where to connect. With the ethereum based payments method, users which host the push notification function will be paid. For the owner to reduce the cost of hosting, the user can host functions.

4.0 Description of Personal and Facilities

4.1 Member and Contributions

4.1.1 Component 1

4.1.1.1 Finding an efficient mechanism to explore the possible ways of bi-directional communication between private and public networks to accomplish the following

Main objectives
To decrease latency of data communication between the nodes.
To improve communication between nodes in the same network efficient
Determining secure ways to store functions in nodes

4.1.1.2 Finding out a mechanism to revert / edit a function deployed in the network
4.1.1.3 Finding out the possibility of creating a new protocol to transmit data between the private and public networks

4.1.2 Component 2

4.2.1 Creating a deployment framework to deploy functions and their configuration in a way to suite the decentralized network.
Security
4.2.2 Packaging
4.2.3 Configuration
4.2.4 Obfuscating functions
4.2.5 Finding out a mechanism to determine a common access point (an endpoint) to call functions deployed in to the network.

4.2.6 Finding out a mechanism to route database a scenario where a single node goes done

4.2.7 Method to retrieve a list of functions to be hosted from super nodes

4.1.3 Component 3

4.3.1 Finding out a proper mechanism to boot up and scale the decentralized network.

4.3.2 Decentralized DNS resolution

4.3.3Finding out the possibility of using super nodes as DNS servers itself

4.3.4 Finding an ideal mechanism to cache or persist data in super nodes

4.3.5 Finding a mechanism to persist data in client application

4.3.6 Finding out a mechanism to replicate the same function in multiple nodes but the replicated function should act as a single function

4.1.4 Component 4

4.4.1 Payment Model based on requests served by a node, amount of data transferred Via smart contracts

4.4.2 Finding out the possibility of paying the nodes which hosts functions based on served requests

4.4.3 Finding out a mechanism to validate the requests served by a mechanism between the origin of the request and the requests served

4.4.4 Finding out a secure mechanism to prevent DoS attacks which lets a node earns money by sending pings.

4.4.5 Creating a portal to deploy smart contracts which works as price models

4.4.6 Finding a method to determine the value of a smart contract.

4.4.7 Deploying a smart contract to the network with a start price

4.2 Time Frame

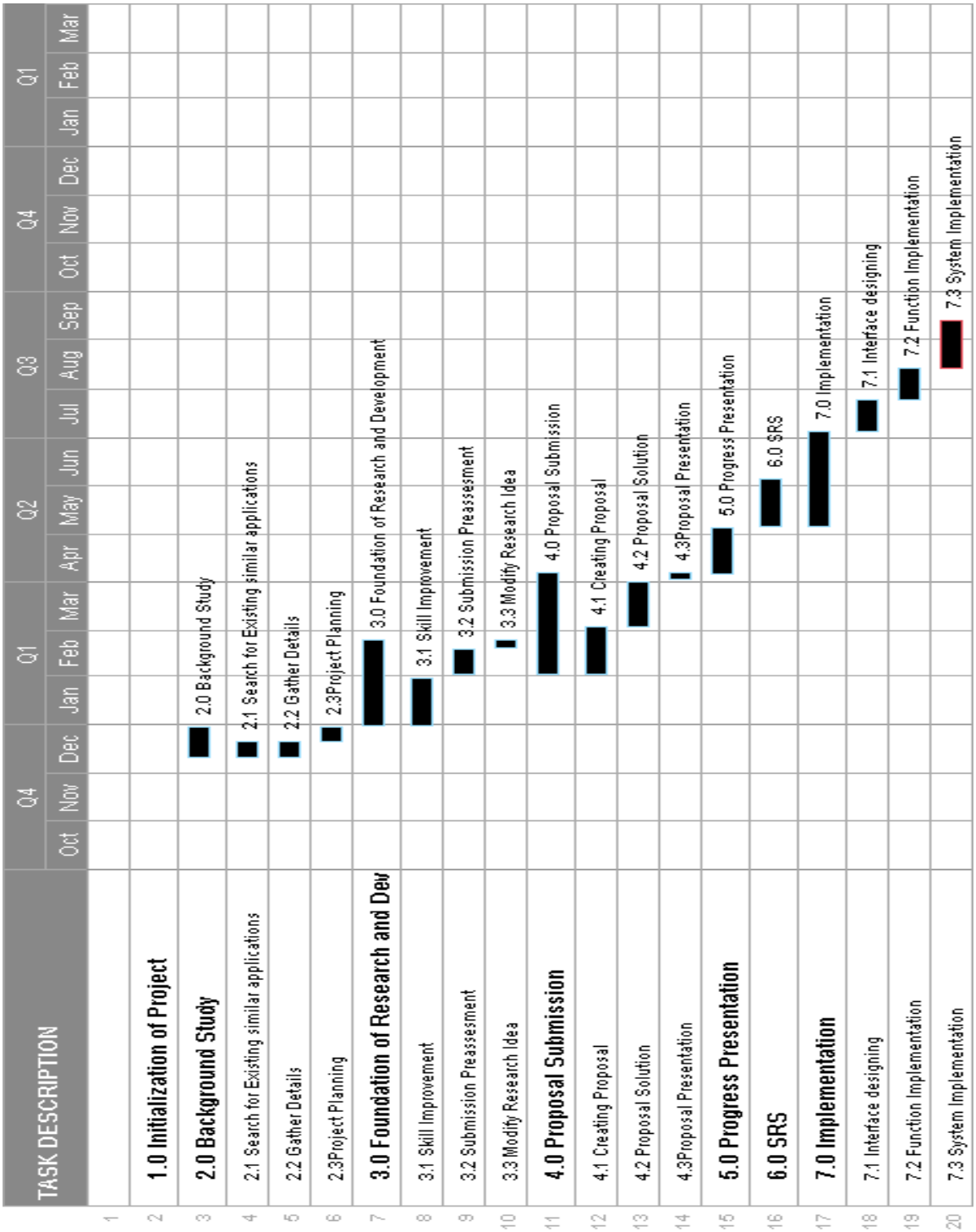


Figure 7 - Gantt Chart

4.3 Work Breakdown Structure

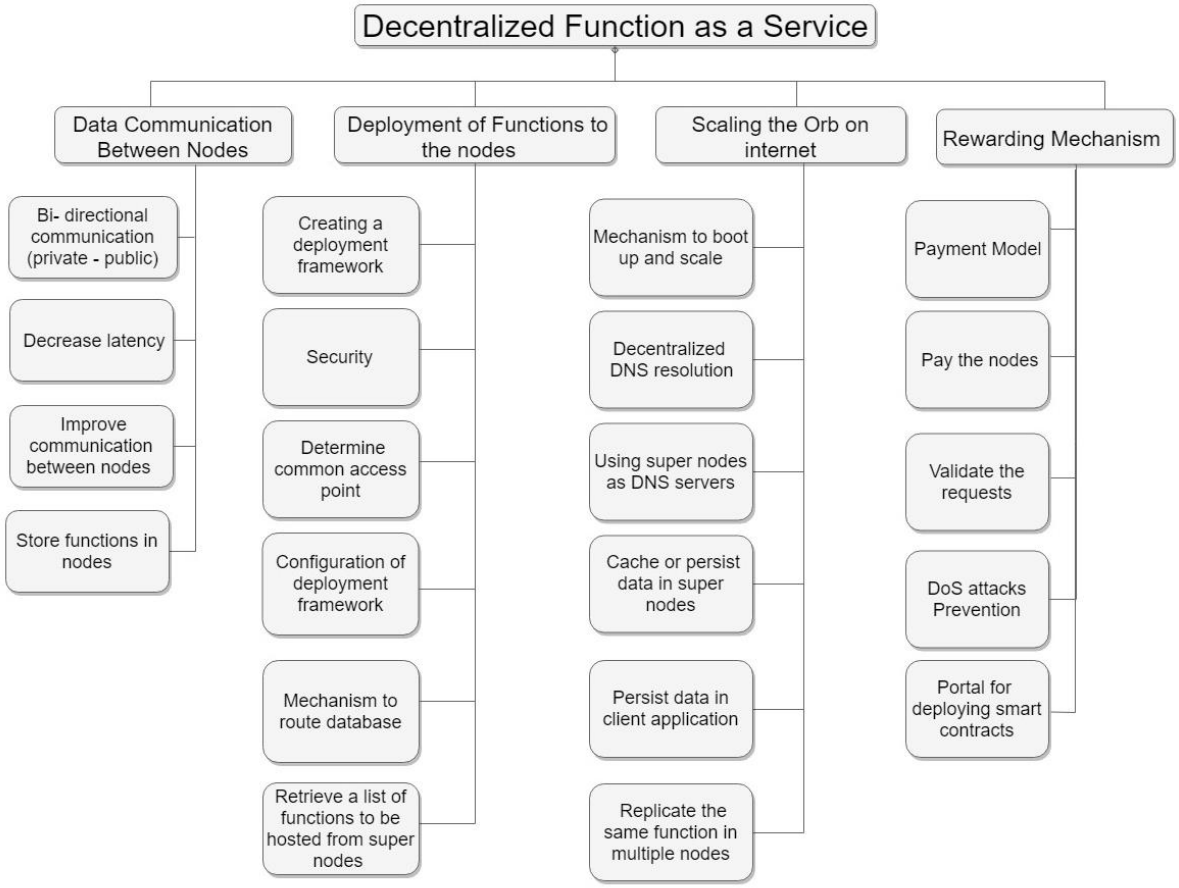


Figure 8 - Work Breakdown Structure

5.0 Abbreviations

Abbreviation	Description
BaaS	Backend as a Service
FaaS	Function as a Service
IPFS	Inter Planetary File System
DNS	Domain Name System
DHT	Distributed Hash Table (network storage)
P2P	Peer to Peer
DDoS	Distributed Denial of Service
IBM	International Business Machines
GFLOPS	Giga-Floating Point Operation
BLOB	Binary Large Object
HTTP	Hypertext Transfer Protocol
REST	Representational State Transfer
STEEM	Science Technology Entrepreneurship Engineering and Math

6.0 References

[1]D. Etherington, "Amazon AWS S3 outage is breaking things for a lot of websites and apps", *TechCrunch*, 2018. [Online]. Available: https://techcrunch.com/2017/02/28/amazon-aws-s3-outage-is-breaking-things-for-a-lot-of-websites-and-apps/. [Accessed: 16- Jan- 2018].

[2]M. Roberts, "Serverless Architectures", *martinfowler.com*, 2018. [Online]. Available: https://martinfowler.com/articles/serverless.html#unpacking-faas. [Accessed: 14- Jan- 2018].

[3]"What is Ether", *Ethereum.org*, 2018. [Online]. Available: https://www.ethereum.org/ether. [Accessed: 18- Jan- 2018].

[4]"How Do Ethereum Smart Contracts Work? - CoinDesk", *CoinDesk*, 2018. [Online]. Available: https://www.coindesk.com/information/ethereum-smart-contracts-work/. [Accessed: 14- Jan- 2018].

[5]"ConsenSys – Medium", *Medium.com*, 2018. [Online]. Available: https://medium.com/@ConsenSys. [Accessed: 21- Jan- 2018].

[6]"What Is AWS Lambda? - AWS Lambda", *Docs.aws.amazon.com*, 2018. [Online]. Available: https://docs.aws.amazon.com/lambda/latest/dg/welcome.html. [Accessed: 22- Jan- 2018].

[7] “Steemit,” *Steemit*. [Online]. Available: https://steemit.com/. [Accessed: 24-Mar-2018].

[9]"AWS Rates Highest on Cloud Reliability", *EnterpriseTech*, 2018. [Online]. Available: https://www.enterprisetech.com/2015/01/06/aws-rates-highest-cloud-reliability/. [Accessed: 14- Mar- 2018].

[10]"The AWS Outage: The Problem with Internet Centralization | Mondo", *Mondo*, 2018. [Online]. Available: https://www.mondo.com/aws-outage-internet-centralization-problem/. [Accessed: 14- Jan- 2018].

[11]"Single Point of Failure – What is it? Why it Matters… :", *Renovodata.com*, 2018. [Online]. Available: http://www.renovodata.com/blog/2015/06/10/single-point-of-failure. [Accessed: 14- Jan- 2018].

[12]"www.ey.com", *Ey.com*, 2018. [Online]. Available: http://www.ey.com/Publication/vwLUAssets/EY-implementing-blockchains-and-distributed-infrastructure/$FILE/EY-implementing-blockchains-and-distributed-infrastructure.pdf. [Accessed: 14- Jan- 2018].

[13]P. Labs, "IPFS is the Distributed Web", *IPFS*, 2018. [Online]. Available: https://ipfs.io/. [Accessed: 14- Jan- 2018].

[14]"Storj - Decentralized Cloud Storage", *Storj - Decentralized Cloud Storage*, 2018. [Online]. Available: https://storj.io. [Accessed: 14- Jan- 2018].

[15]*Google Cloud Platform Documentation | Documentation | Google Cloud*. [Online]. Available: https://cloud.google.com/docs/. [Accessed: 02-Feb-2018].

[16]"Kubernetes Documentation," *Kubernetes*. [Online]. Available: https://kubernetes.io/docs/home/?path=users&persona=app-developer&level=foundational. [Accessed: 02-Feb-2018].

[17]"Decentralized Internet on Blockchain – Hacker Noon," *Hacker Noon*, 03-Oct-2017. [Online]. Available: https://hackernoon.com/decentralized-internet-on-blockchain-6b78684358a. [Accessed: 02-Feb-2018].

[18]*Blockchain Based Distributed Control System for Edge Computing - IEEE Conference Publication*. [Online]. Available:

http://ieeexplore.ieee.org/abstract/document/7968630. [Accessed: 02-Apr-2018].

[19]"Decentralizing Your Microservices Organization," *The New Stack,* 31-Oct-2017. [Online]. Available: https://thenewstack.io/decentralizing-microservices-organization/. [Accessed: 02-Apr-2018].

[20]"Building Your First Network," *Building Your First Network - hyperledger-fabricdocs master documentation*. [Online]. Available: http://hyperledger-fabric.readthedocs.io/en/release-1.0/build_network.html. [Accessed: 02-Apr-2018].

[21]M. Meister, "leveraging Kubernetes to run a private production ready Ethereum network," *Medium,* 26-Nov-2017. [Online]. Available: https://medium.com/@cryptoctl/leveraging-kubernetes-to-run-a-private-production-ready-ethereum-network-b6f9b49098df. [Accessed: 02-Apr-2018].

[22]*A Decentralized Service Discovery Approach on Peer-to-Peer Networks - IEEE Journals & Magazine*. [Online]. Available: http://ieeexplore.ieee.org/abstract/document/5928313/. [Accessed: 02-Apr-2018].

[23]"Taming WebRTC with PeerJS: Making a Simple P2P Web Game," *Toptal Engineering Blog*. [Online]. Available: https://www.toptal.com/webrtc/taming-webrtc-with-peerjs. [Accessed: 02-Apr-2018].

[24] "Decentralized payments for environmental services: The cases of Pimampiro and PROFAFOR in Ecuador," Ecological Economics, 31-Dec-2007. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0921800907005320. [Accessed: 05-Jan-2018].

[25] A. Crespo and H. Garcia-Molina, “Semantic Overlay Networks for P2P Systems,” *SpringerLink*, 19-Jul-2004. [Online]. Available: https://link.springer.com/chapter/10.1007/11574781_1. [Accessed: 02-Mar-2018].

[26] “Rethink the Web browser,” Beaker | Peer-to-peer Web browser. No blockchain required. [Online]. Available: https://beakerbrowser.com/. [Accessed: 05-Feb-2018].