“ORB”

DECENTRALIZED FUNCTION AS A SERVICE (DFAAS)

Project ID – 18-070

Preliminary Progress Review Report

R.G.D Nayomal	IT15027948
S.K.N.U Tissera	IT15120212
T.H.A.K Silva	IT15136916
A.T.Nimansa	IT15419910

Bachelor of Science Special (Honors) Degree in Information Technology

Department of Software Engineering

Sri Lanka Institute of Information Technology

Sri Lanka

March 2018

Table of Contents

Table of Figures

1. Introduction

1.1 Purpose

The purpose of this document is to provide the details of the overall scope, overview, requirements, functional design, key research methodologies and the main deliverables of the Research, Decentralized Functions as a Service. This document will elaborate the purpose and a complete declaration for implementing a prototype, including the explanations of background, research constraints, deliverables or outcomes, anticipated benefits. This document is primarily intended for the developers of the platform and the consultants to clarify the requirements, processes and constraints.

1.2 Scope

1.2.1 Bi-directional Data communication between Nodes

Main objectives of this research component is to explain how todecrease latency of data communication between the nodes and the super nodes, function consumption, editing, reverting and securely distribution. Finding mechanisms to reduce the node discovery times depending on the network, the nodes reside. A Research constraint will be to overcome the communication between nodes, super nodes behind NATs. The possibility of creating a new protocol to transmit data between the private and public networks is explored. Also the data communication must be secure, therefore, usage of secure web sockets for data communication is defined. The mechanisms to consume a function, edit, roll back a function and to distinctively identify a function from the user is a challenge which is also elaborated. The clients should be able to consume a function deployed into the network by initially establishing a connection to the network. In order to achieve function consumption, the client must be able to find the hosted nodes the network and execute the function from the nearest possible node with lower latency to increase the efficiency. The

deployed function data must be stored initially in the client node, super nodes and seeding node respectively. The process which will be used to determine seeding and deployment is also elaborated in this section. The current progress of the research prototype is experimental, therefore the minimum requirements that is needed for Orb to move from experimental to the production level needs fine tuning of the inner workings of both client and server modules. This will be elaborated more in the detailed design section.

1.2.2 Deployment framework to deploy functions

Deployment framework is built on top of node infrastructure at client end to facilitate hosting and seeding functions. This resides behind the user and act as a wrapper for the API. Client Software is the GUI for the client. This facilitates from Registration of the user to Deploying a function. Steps include; authenticating user in the decentralized network, logging to the system, writing a function, obfuscate the function, packaging and deploying a function, providing access tokens for the specified function. Main purpose of this is to bridge the gap between client and network infrastructure.

Security is maintained throughout the system from registration of user to function deploying and calling. Although this is a decentralized system, security is good than a centralized network. Decentralized network is proof of DDoS attacks since all requests are routed in the system via supernodes. Socket.io plays a role is securing network proofing external attacks.

Packaging is done for security purpose and easy of sharing. JavaScript function is packaged (using a node library called ‘pkg’) before deploying.

Obfuscation is the process of making something difficult to understand. JavaScript function is obfuscated because it’s an intellectual property and it prevent an attacker from reverse engineering function in the network. It normally renames useful class and variable names to meaningless labels or adding unused or meaningless code to an application binary. The output after obfuscating is harder to read and understand.

1.2.3 Booting up and Scaling Orb

This will give a basic idea of key components that are existing in the platform, how they interact which each other, how system initially boots up and communication mechanism of the platform network architecture underneath the real architecture and to address the problem of communication gap between public to private network and build bridge between them.

Initially when the system boots up, there are only known super nodes. When user install client application client application know what are super node are existing in network. These nodes are connecting socket streams. Which allow platform to communicate bi-direction. When a client gets registered in the system, Super Node identify it as a node. Node is identified using unique Orb id. Even after every node addition, Super Node populates its data which is saved in a DHT throughout the network. This makes the transparency between Super Node like nodes feel it as a central computer.

1.2.4 Rewarding System

Scope of the Rewarding system component is a collection of several activities. Payment Model based on requests served by a node. The functionalities our rewarding system should possesses includes the following sections.

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

The function owner initially hosts his own function and if some user is interested in hosting it, he can seed it to be used by requesters. When requester, request for a specific function the function owner and the seeder both gets rewarded as per the requests for a pre-defined ratio.

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

Validating the requests mainly focuses on blocking, function owner or seeder to send requests to its own function and corrupt the rewarding system. For the above-mentioned function finding out a secure mechanism to prevent DoS of attacks which lets a node earns money by sending pings should be done.

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

A smart contract is a computer protocol intended to digitally facilitate, verify, or enforce the negotiation or performance of a contract. Smart contracts allow the performance of credible transactions without third parties. These transactions are trackable and irreversible. This is needed when we need to distribute the rewards to the users. We can implement the logic of price models via the smart contract and distribute the rewards.

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

The nodes might host different kinds of functions with different complexities. It is not fair to treat each and every function the same and reward equally. There should be a method to estimate the complexity of the function and calculate the value of the function accordingly

1.3 Definitions, Acronyms, and Abbreviations

PPR	Preliminary Progress Review
DoS	Denial of Service
GUI	Graphical User Interface
DHT	Distributed hash table
DDOS	Distributed Denial of Service
NAT	Network Address Translators
FaaS	Functions as a Service
IPFS	Interplanetary File System
DFaaS	Decentralized Functions as a Service
API	Application Programming Interface
AWS	Amazon Web Services
EC2	Amazon Elastic Compute Cloud
SaaS	Software as a Service
AWS ECS	AWS Elastic Container Service
IoT	Internet of Things
Orb	Name of the research prototype
Seed	A node which hosts a function
Function	A executable code which performs a well-defined specific function
pkg	A JavaScript interpreter which executes code in a sandbox
Sandbox	A safe to use testing environment
Google App Engine	A SaaS offered by Google, where a user can deploy web services to the Google cloud automatically

1.4 Overview

The Intention of Orb is a Decentralized Functions as a provider consisting of a network of decentralized Nodes and Super Nodes which acts as the main infrastructure of Orb. On top of the infrastructure, the collection of nodes and Super Nodes, a framework is built to let the users consume the services, contribute to the services and to obtain Rewards. Rewards are the key factor to attract the users into the network by allowing the network to scale rapidly.

1.4.1 Main Goals

The main goals of this research is to create an ecosystem of decentralized Services which helps to eliminate the need of centralized APIs. Therefore, a developer can keep focus on the functionality of the application rather than focusing on the deployment since Orb automatically takes care of the deployments. The secondary goal is to utilize idle consumer computing power and to Reward the contributors in terms of seeding, consumption and deployment of functions through a decentralized Rewarding System.

1.4.3 Major Functionalities and Tasks

There are 3 Major Tasks or Functionalities where a particular user could interact with the Orb. Deployment of Functions to the Orb network being the major functionality, Seeding of a function and Consumption of a already deployed function could also be achieved.

2. Statement of Work

2.1 Background information and overview of previous work based on literature survey

The area of Decentralized applications has gained traction recently. With static content sharing networks such as IPFS, and browsers such as beaker browser, mist, 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 with a unique Rewarding system, the users are able to deploy Serverless functions which could be used as services in a more decentralized internet. The idea of services being decentralized is unique. Therefore, there were no direct similar work which is equivalent to Orb but there were systems built focusing on the decentralization, Reward Distribution, Static content Distribution and Torrents. The commercialization of the orb is achieved by attracting users to use the network as service providers or service users.

2.1.1 AWS Lambda

Amazon web services lambda is also providing server space for the functions to be deployed. With AWS Lambda[8], it's possible to 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. Comparing Orb to Lambda is impossible since the concept of Lambda is more centralized where as Orb is fully Decentralized. But the idea of focusing more towards the service itself rather than relying on background is common to both AWS Lambda and Orb.

2.1.2 Steemit

Steemit[9] is a social network and a content rewards platform that makes the crowd the beneficiaries of the attention economy. It does this by 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. More similar principals are used in Steemit Rewarding System and Orb but Rewards are for two different reasons. Steemit focuses more on Rewarding users that have written, contributed and read Steemit articles whereas Orb focuses more on the deployment, contribution, seeding and consumption of functions.

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

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.

2.1.3.2 IPFS Objects

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

- Data - a BLOB of unstructured binary data of size < 256 kB.
- Links - an array of Link structures. These are links to other IPFS objects.
- A Link structure has three data fields:
- Name - the name of the Link.
- Hash - the hash of the linked IPFS object.
- Size - the cumulative size of the linked IPFS object, including following its links.

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.

2.2 Identification and significance of the problem

Today, most of the web services are built around centralized client server architecture. With the dawn of Serverless computing, usage of Serverless functions have gained traction. In fact, many colorations have integrated Serverless functions to their core web services. The main advantage of using Serverless functions is that the developers can focus more towards the implementation of the function rather than integrating the function with a complex API just to provide an endpoint for consumption. Serverless functions save time, energy and cost. But these systems comes with few drawbacks as mentioned below.

2.2.1 Drawbacks of Centralized Systems

2.2.1.1 Single Authority

AWS provides EC2 service which is used to create virtual machines. The users can easily deploy web services/applications to EC2. EC2 has built in auto-scaling and have higher reliability (availability rating of 99.9974 percent, Cloud Harmony 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%), 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 won't be able to access 40% of the web services. It's analogous to a massive breakdown of the internet services, similar to the massive internet outage happened on February 28^{th,} 2017. Decentralization eliminates the single authority problem

2.2.1.2 Data Owned by 3rd Parties

Deploying a function to a third-party services provider always has its drawbacks. The organization automatically gets the ownership and access to the data. Although service providers like AWS could be trusted, there is always a risk. In a decentralized system, data is stored in a collection of nodes. Therefore, a single entity won't be able to control how the data is stored in the system.

2.2.1.3 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. This problem persists in a small or medium-sized business where the entire business logic is hosted in a single centralized server. Since that business might run low on profit affording for cloud computing / AWS will hardly be possible. Decentralized systems doesn't have this problem.

2.2.1.4 Most Systems are not scalable by default

Most cloud services serve a large number of clients, thereby generating log of API load. In order for a system to handle traffic continuously and to serve for incrementing demand, there must be a way to scale up the system horizontally and vertically. Vertical Scaling includes hardware upgrades done to the server which increases the cost of hosting the web service. Horizontal scaling is achieved by deploying load balances which proxy requests into few identical instances running the same web service. Running parallel instances cost the user. Generally in AWS, features like horizontal scaling must be setup of manually. AWS ECS and google App Engine SaaS supports

horizontal scaling by default. But in the decentralized system, adding more nodes will automatically scale up the system effortlessly.

2.2.1.5 Centralized Systems Generate More Traffic

Since all the business logic implemented as a web service is aggregated in a central location, all the clients calling the same instance in parallel increases the load average of the server, if not load balanced, which is another configuration overhead. Not adhering to best practices such as load balancing cloud lead to a system failure as mentioned in 2.2.1.3 sub section. Decentralization of the system could prevent from the traffic generation problem since the service is fully decentralized over a set of nodes.

2.2.1.6 Security of Data

Due to the centralized nature of the system, the system is vulnerable to attacks such as DoS, DDoS, Man in the middle attack and hacking. There are security measures like https, end to end encryption which can provide some security against the attacks. But decentralizing the system would eliminate DoS, DDoS and man in the middle attacks completely.

2.2.2 Drawbacks of Serverless Functions

2.2.2.1 The function is accessed through a 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.

2.2.2.2 Lack of Operational Tools

The developers are dependent on vendors for debugging and monitoring tools. Debugging distributed systems is difficult and usually requires access to a significant amount of relevant metrics to identify the root cause.

2.2.2.3 Architectural Complexity

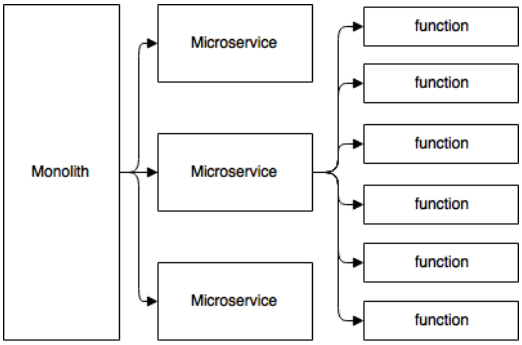

Figure 1: Architectural Complexity

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.

Decisions about how small (granular) the function should be, takes the time to assess, implement and test. There should be a balance between the number of functions should an application call. It gets cumbersome to manage too many functions, and ignoring granularity will end up creating mini-monoliths.

AWS Lambda, for now, limits you to how many concurrent executions you can be running of all your lambdas. The problem here is that this limit is across your whole AWS account. Some organizations use the same AWS account for both production and testing. That means if someone, somewhere in your organization does a new type of load test and starts trying to execute 1,000 concurrent Lambda functions, you'll accidentally DoS your production applications.

2.2.3 Computing Power and Efficiency

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.

2.2.4 Solution

Orb resolves almost all the problems mentioned above. The most problems of centralized systems are eliminated due to the native decentralized nature of the system. As of today, Orb is the worlds' first Decentralized Function as a Service provider. There are no similar systems which could be directly compare with the Orb. There are 3 main sections that are unique in Orb

I. Decentralized network of peers - Infrastructure
II. Decentralized Function as a Service - Service
III. Decentralized Rewarding system – Rewards

Using Functions as a Service provider like AWS Lambda is costly compared to the Orb including features like auto scaling. But with Orb, auto scaling comes natively and deploying could be easily done through the provided client application.

Many existing systems like Steemit has attractive payment models. But there are no instances in the history of using a decentralized Rewarding System to reward the contributors, seeders and consumers of the Function.

2.3 Technical objectives

This need some hardware and software to perform. As software requirements it needs client software. Client software is available for any platform. Client software is made using electron framework[29]. It's supporting cross platform.

This system needs a computer with 4Gb ram and i3 processor as minimum requirements. Consumer can use a desktop or laptop as a device.

2.4 Detail design

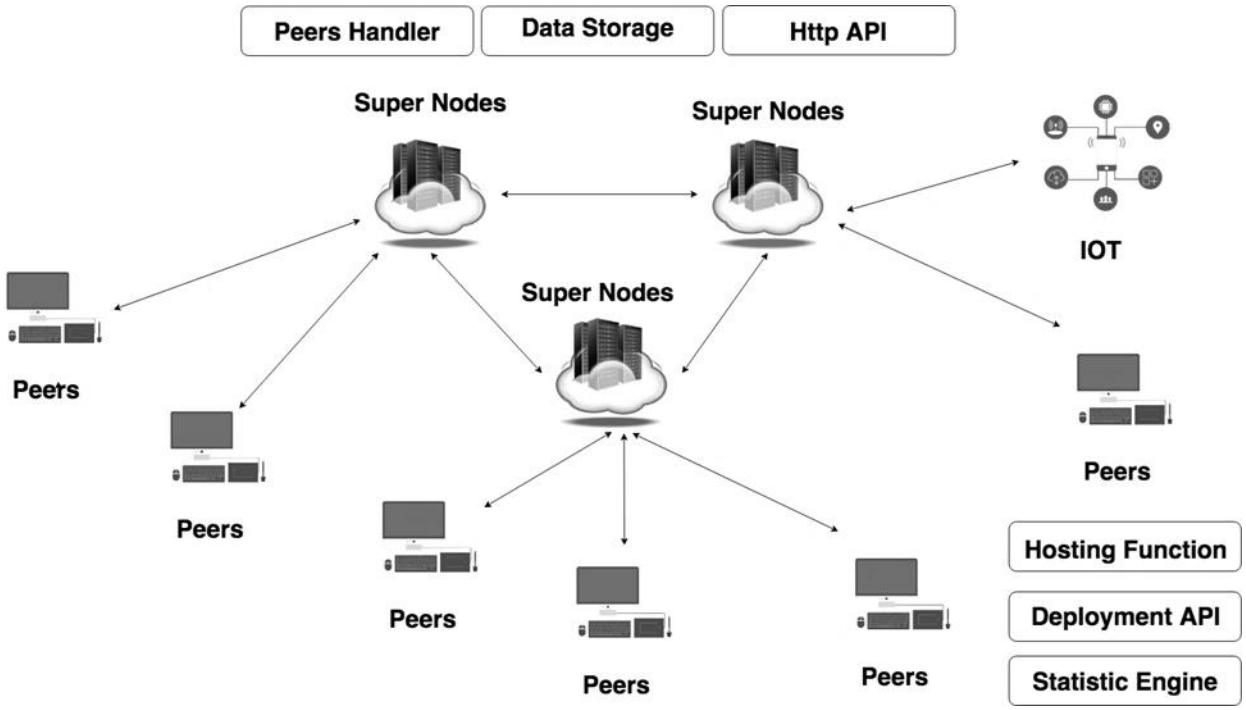

Figure 2 : High-level Architecture

2.4.1 Network Infrastructure

This system composed of both software and hardware requirements. Orb relies on a p2p network, composed of nodes and super nodes represented by 1 and 2 respectively according to the figure shown below. The nodes act as clients and super nodes act as mediators which helps node discovery. When the number of nodes increases, there must be a sufficient number of super nodes to route requests. Otherwise the network will be flooded. A mechanism to promote node to a Super Node by obtaining a public IP address to a node is under discussion. If the network supports automatic promotion of the nodes to deal with the incoming traffic, the resiliency of the network will increase in vast amount.

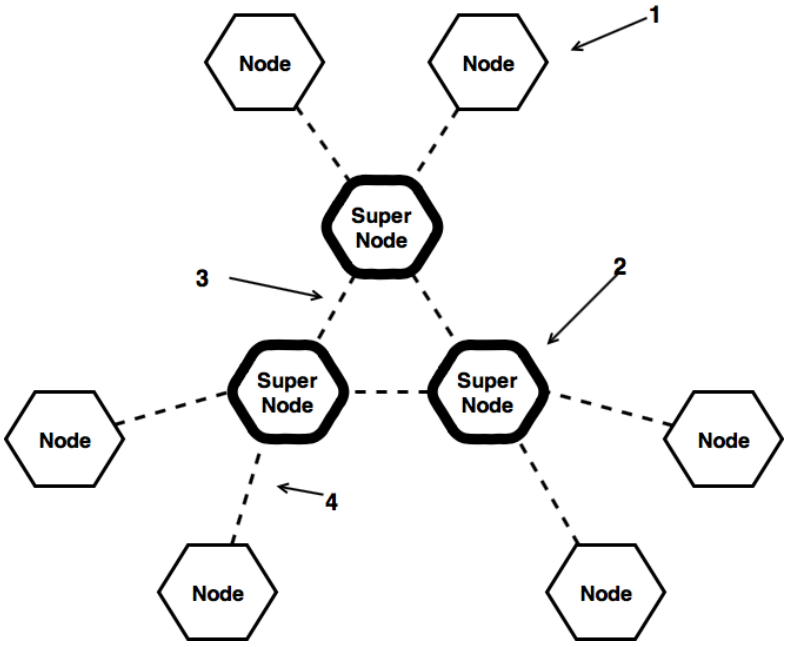

Figure 3: Network Infrastructure

2.4.2 Basic Architecture of a Super Node

The super nodes are composed of Function Distribution Framework and the Rewards framework. Following is a block diagram where the node is represented by 5.

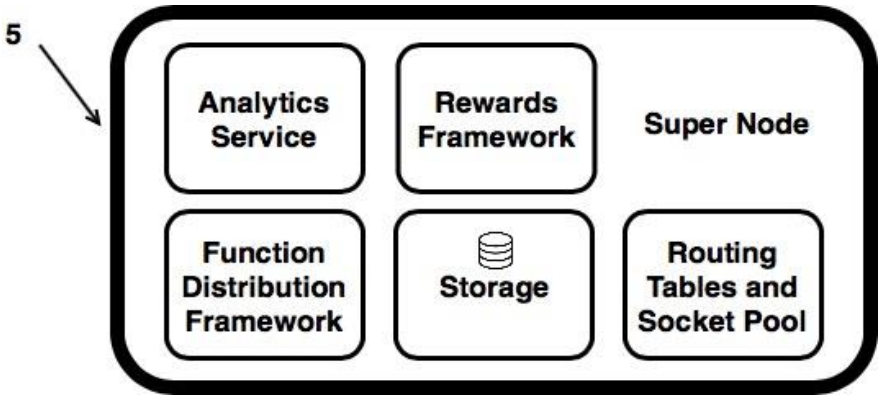

Figure 4 : Basic Architecture of a Super Node

2.4.3 Node to Supernode Communication

The Super Nodes always contains a public IP address which is known to nodes. Therefore, a node connects to a super node at the start of the network and though the super node, it discovers other super nodes and nodes and establishes connections accordingly as shown by 4 in the Fig [3]. The nodes follow the principles of "Satoshi node discovery" methodology. The node discovers the IP address and port of super nodes in several different ways. Orb nodes uses a predefined set of known super nodes stored in the internal data storage of Orb Deployment Framework to initiate a connection with the super node. Then the node retrieves an access token from the super node along with the super node socket id which will be stored in the client software and uses for further communication with the respective supernode. Given below is the connection flow which occurs between a node and a super node at the initiation of the Orb network.

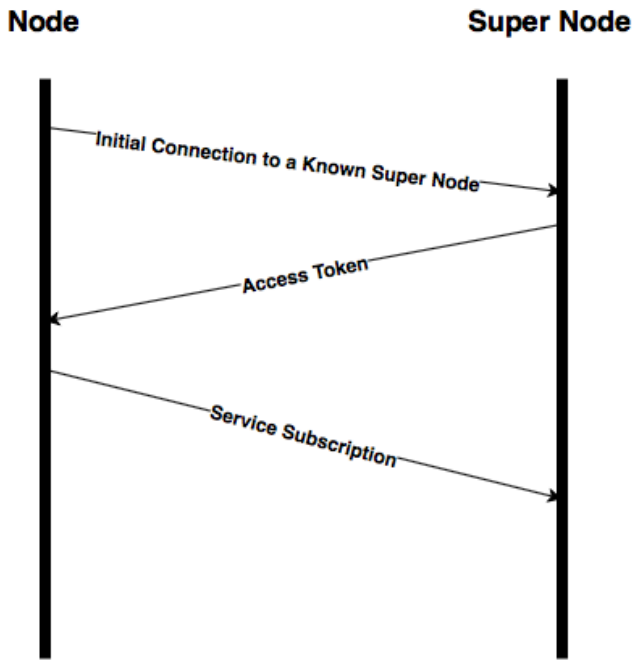

Figure 5: Node to Supernode Communication

2.4.4 Supernode to Supernode Communication

Super nodes are predefined reliable set of nodes specialized to mediate requests. Initially the super nodes finds other super nodes similarly by keeping a set of known super nodes in its internal storage. This way, a super node gradually establishes connections with the remaining super nodes. A timestamp is kept for each address to keep track of when the node address was last seen. The Address Currently Connected handles updating the timestamp whenever a message is received from a node. Timestamps are only updated on an address and saved to the database when the timestamp is over 20 minutes old. Given below is the connection flow which occurs between super nodes at the initiation of the Orb network.

2.4.5 General Functionality

2.4.5.1 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.

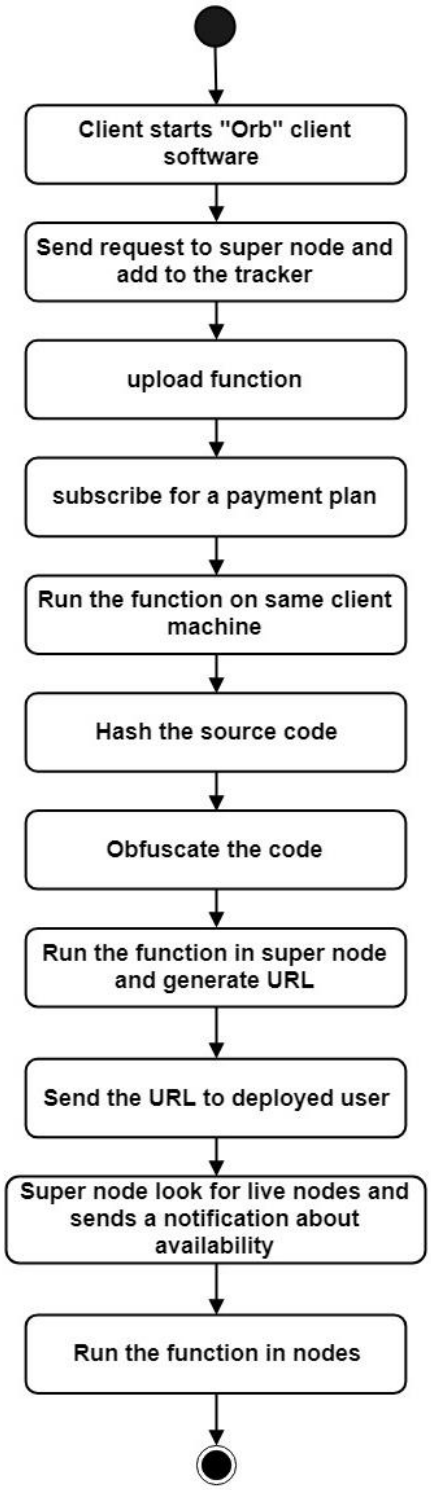

Figure 6 : General Functionality

2.4.6 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.

2.4.7 Sample Scenario

2.4.7.1 Global Temperature warning system using IoT

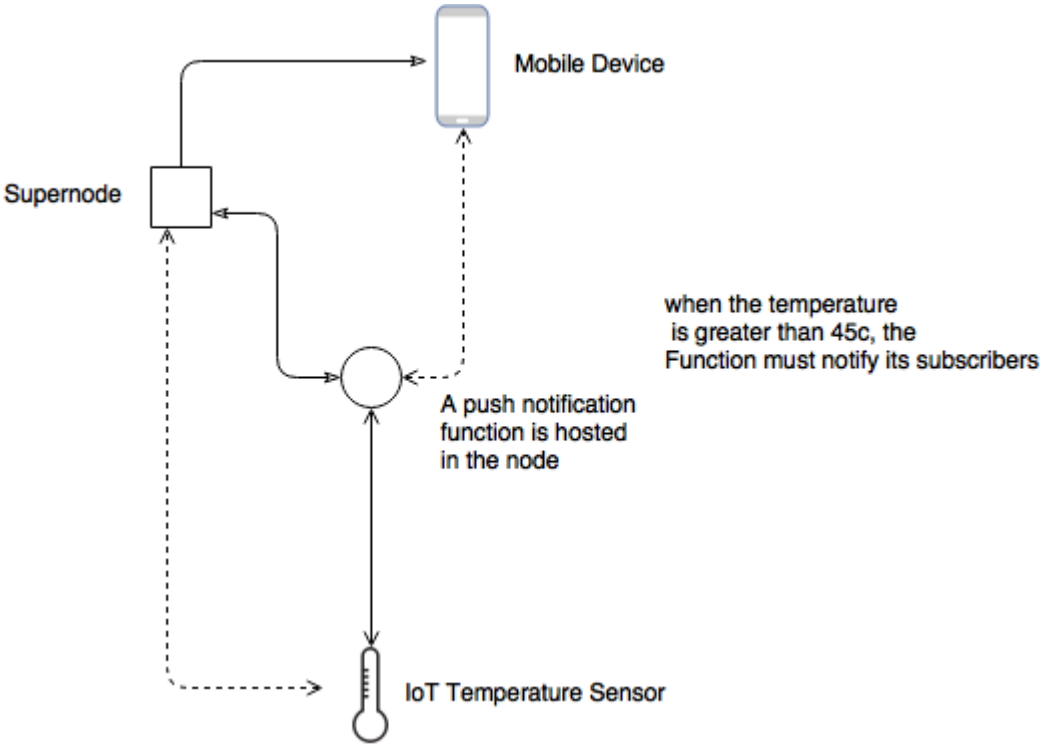

Figure 7- How Orb helps to host a temperature function

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.

2.5 Sources for test data & analysis

As our system is a system which provide deployment space to its users so there are two kinds for users who input and there are two groups of input types. They are

I. Inputs from function owners – function and updates
II. Inputs from function requestors – these differ according to the function they use. If we take image processing function as the input it will act as follows

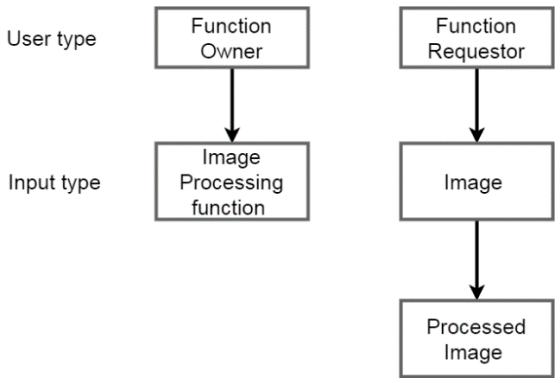

Figure 8: Sources for test data & analysis

When the function owner uploads a function the system should be able calculate the requirements to host it and display it to the seeders. Once the function is up there should be tests to check whether the function will be available 24/7 without crashes.

After function deployment is completed the reachability of the function to the function requestor should be validated. When a function requestor uses a function we should check whether it will provide what it has agreed to give and should remove all the functions that are developed to corrupt the system and gain profits by fraud.

Input	Things to test
Function	Get the required space and check whether function can run on mobiles, laptops etc Whether function will run when Triggered Whether function will be available without crashes
Inputs for the function (Eg- Image for Image Processing function)	Whether the requester can reach the function. Whether function will execute properly when inputs are given. Whether function will give what the user actually expects.

2.6 Anticipated benefits

2.6.1 Stopping Single authority

There are many drawbacks in current centralized systems. The first main thing is single authority. AWS is a largest cloud services provider which offer many services and they hold a major part of internet. Although most of the organizations use cloud services, they hand over the authority of data to a single service provider, which creates a monopoly within the market.
Orb is a decentralized service provider, hence there will be no single set of servers that own all the data in the world and Orb allows the organizations who uses it to give the ownership of data back.

2.6.2 Stopping Single Point of Failure

Cloud outage is inevitable. It can happen anywhere, anytime. There are many incidents where single point of failure made great losses to the companies who uses cloud services. When every resource is in a centralized manner DOS attacks and security threats will rise more and more.
Orb is designed to act in a decentralized manner so the risk of having downtimes due to a centralized architecture will be minimized. The user will be able to enjoy a reliable service with a better experience.

2.6.3 Cost Saving

When a function or a system is deployed in a cloud, and when the requests goes high or it scale higher they server space won't be enough for that scarce resources the user will have to pay more. The cost will be too high when auto scaling happens
In Orb scaling is not a problem at all. There are many peers ready to be seeders since they get a gain out of it and the node who is deploying the function in to orb will be benefited as well. Through this method the cost will be reduced and will be able to get additional rewards too.

2.6.4 Stops Vendor locking

the problem in many centralized networks. It is a situation in which a customer using a product or service cannot easily transition to a competitor. Vendor lock-in is usually the result of proprietary technologies that are incompatible with those of competitors. However, it can also be caused by inefficient processes or contract constraints, among other things. When a function is deployed to Orb, it will be hosted all over the peers which avoids the disadvantages of vendor locking

2.6.5 Anonymity

Orb does not use any of the real personal information to create the Orb account for a user. Hence anyone can upload a function and anyone can request for functions anonymously. Being anonymous prevents hackers from getting access to sensitive information such personal data, credit card transactions, passwords, and banking information. With all these advantages, it's easy to see why most Internet users value online anonymity.

2.6.6 Gets Rewarded when contributed

Orb has 3 main types of users and from these types function owners and function seeders are the contributors who gain rewards. Function owner and the seeders get rewarded as per the requests the function gets. Function owner get paid if the function get requested by other users for 10^6 times (in 2:8 proportion with seeders)

2.6.7 Function over Configuration

Orb user only needs to focus on the functionality of the function he is going to deploy. The configurations and all the complex procedures are handled by the Orb client software. There is no need of separate API gateways

3. Project plan and schedule

3.1 Time frame

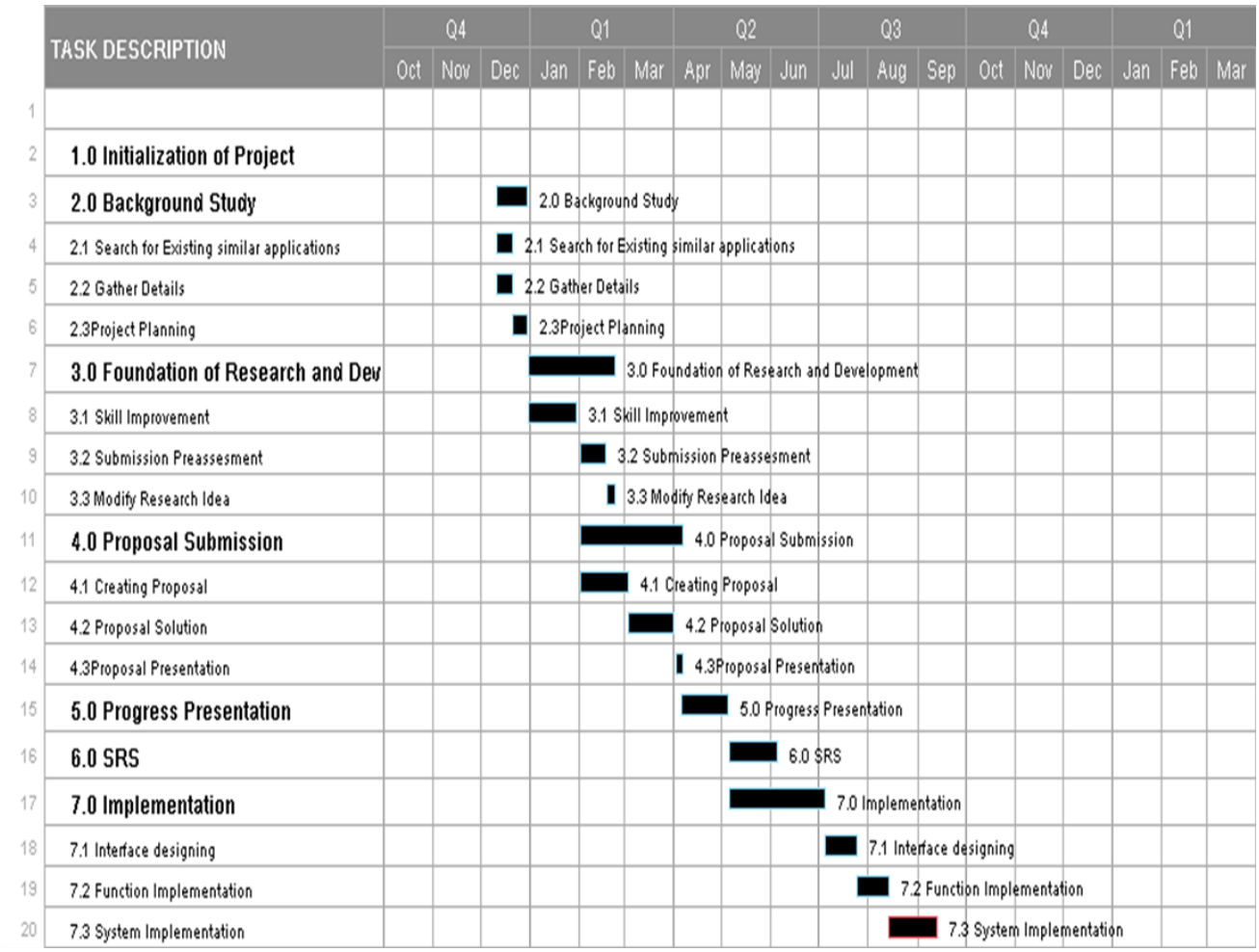

Figure 9- Gantt Chart

3.2 Work Breakdown Chart

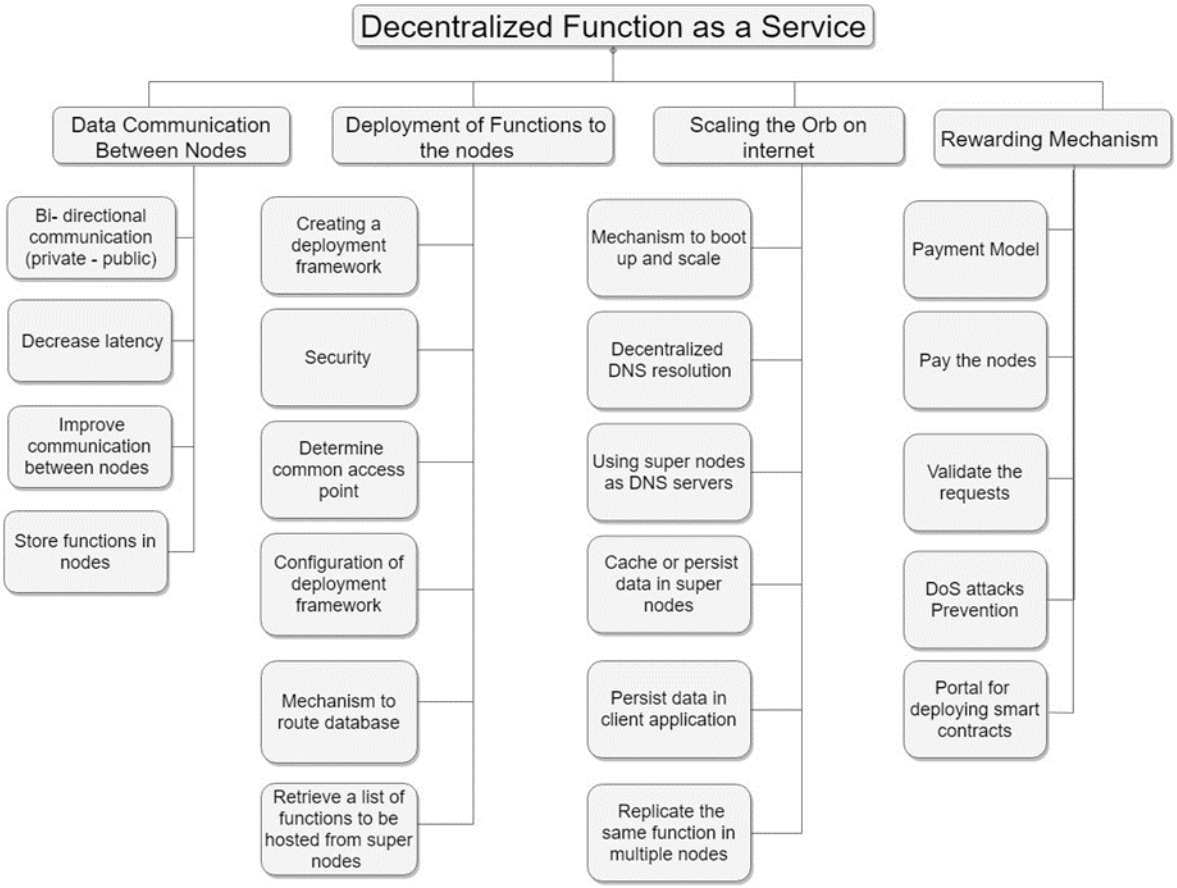

Figure 10-WBS

4. Research constraints

4.1 Lack of availability

This type of Decentralized system may lead to the lack of availability for a function at the level of less scaled network. Functions hosted in the Orb might suffer from availability issues if the number of nodes seeding the function become less. But this problem may persist until Orb reaches a considerable number of nodes. Before that system is less scaled. In a level like that availability of a function may be lower.

4.2 Language dependent

At this level of startup, we can only write functions in JavaScript. That a considerable constraint since there are more popular languages in trend as JavaScript. (e.g.: Java, Ruby, Scala, C++, C, Objective C, C#, python etc.). Orb will support others in future.

4.3 Inability of Bootup without 2 nodes

This requires minimum of 2 nodes and one supernode to bootup the system. But in Other centralized systems it's not needed. It only needs one machine. So in here it's constraint.

5. Specified deliverables

This has few software and hardware deliverables. So, let's discuss about software deliverables first.

I. Client software which makes the user easy to interact with the system. Hence, increase transparency to the user.

This is the Graphical user interface (GUI) for the client. This facilitates from Registration of the user to Deploying a function. Steps include; authenticating user in the decentralized network, logging to the system, writing a function, obfuscate the function, packaging and deploying a function, providing access tokens for the specified function. Main purpose of this is to bridge the gap between client and network infrastructure. This includes the following features in it.

I. Create an Orb id
II. Generate a Key-pair as encrypted password
III. Interface for logging to system using Orb id and Key-pair
IV. Code editor for writing functions
V. Inbuilt Obfuscator
VI. Inbuilt Javascript Interpreter
VII. Interface for display available functions

II. Deployment framework to deploy functions

Deployment framework is built on top of node infrastructure at client end to facilitate hosting and seeding functions. This resides behind the user and act as a wrapper for the node. Nodes ultimately communicate via sockets with the Supernodes.

III. Orb Wallet

The deliverable in this section is a software which act as a wallet to the user. Using this wallet the user will be able to track all his activities, get total rewards, get total expenses and let the user have a great user experience with high security and low risk.

Orb wallet is an application that will,

I. Let the user get registered and login to make transactions
II. Show the list of transactions occurred in the past.
III. Show the total reward earnings and deductions
IV. Calculate the deployment cost for the functions that are to be uploaded
V. Notify the seeders and function owners when they get paid.

The Orb wallet will be able to get installed once Orb client software is installed.

IV. Server Software

This is the software installed in the Supernode. It capable of managing all the requests and route in the system. This perform health checks and keep a record on latency of nodes. So when a request comes it can route to the node with lowest latency.

6. References

[1]"What is obfuscation (obfu)? - Definition from WhatIs.com", SearchSoftwareQuality, 2018. [Online]. Available: https://searchsoftwarequality.techtarget.com/definition/obfuscation. [Accessed: 07- May- 2018]

[2]"An Introduction to IPFS – ConsenSys – Medium", Medium, 2018. [Online]. Available: https://medium.com/@ConsenSys/an-introduction-to-ipfs-9bba4860abd0. [Accessed: 07- May- 2018]

[3]"What is a Torrent? - Definition from Techopedia", Techopedia.com, 2018. [Online]. Available: https://www.techopedia.com/definition/5263/torrent. [Accessed: 07- May- 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].

[27] "• Statista - The Statistics Portal for Market Data, Market Research and Market Studies", Statista.com, 2018. [Online]. Available: https://www.statista.com/. [Accessed: 15- May- 2018].

[28] "How Do Ethereum Smart Contracts Work? - CoinDesk", CoinDesk, 2018. [Online]. Available: https://www.coindesk.com/information/ethereum-smart-contracts-work/. [Accessed: 15- May- 2018].

[29] "Electron | Build cross platform desktop apps with JavaScript, HTML, and CSS.", Electronjs.org, 2018. [Online]. Available: https://electronjs.org/. [Accessed: 15- May- 2018].