“ORB”
DECENTRALIZED FUNCTION AS A SERVICE (DFAAS)

Project ID – 18-070

Preliminary Progress Review Report

R.G.D Nayomal

IT15027948

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

1.0 Introduction

1.1 Purpose

The purpose of this document is to describe how the functions are transferred and managed in the the Orb network by the Function Distribution Framework. This document defines the critical processes and methodologies used by the Function Distribution Framework. Furthermore, this document focuses on how the Function Distribution Framework reacts to different scenarios.
This document is highly technical contains patent pending materials and is intended for the Distributed Systems Engineers and Software Engineers.

1.2 Scope

The Function Distribution Framework is created to provide a framework to control how the deployed functions are handled and distributed over the Orb network.

1.2.1 Challenges

The Major Challenge is to distribute functions across nodes located in private networks and handling requests. This document will elaborate the mechanisms used to overcome the challenges and the constraints.

1.2.2 Function Deployments

This section covers the sequence of events that is trigged by a function deployment carried out by a client. When a Function is deployed, the process of notifying the nodes about the function is elaborated.

1.2.3 Routing Requests

This section elaborates how the deployed functions requests are routed and consumed by clients through a SDK.

1.2.4 Function Distribution Methodology

This section elaborates on how the functions are decentralized in an efficient manner.

1.2.5 Security

The security concerns pertaining to the function distribution framework and counter measures deployed to prevent a security flaws are focused in this section.

1.2.6 Mechanisms to Edit and Decommission Function

In a centralized system, editing and decommissioning a function can be easily done by logging to the instance. Since all the functions are decentralized in different nodes, it's difficult to edit or decommission. Therefore, special mechanisms have to be followed. This document will highlight the key mechanisms used to resolve editing and decommissioning of functions

1.3 Overview

Main objectives of Function Distribution Framework is to explain how to decrease 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 is 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.

1.3.1 Main Goals

1.3.1.1 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 Super Node in between where all the communication is done through the Super Node. Implementing the underline server and client software should be done to manipulate requests through the network.

1.3.1.2 Determining secure ways to store functions in nodes finding out a mechanism to revert, edit a function deployed in the network.

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

1.3.1.4 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.

1.3.2 Major Functionalities and Tasks

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 Definitions, Acronyms, and Abbreviations

SDK	Software Development Kit
UI	User Interface
IPFS	Inter Planetary File System
IP	Internet Protocol
NAT	Network Address Translators
REST	Representational State Transfer
API	Application Programming Interface

2 Statement of Work

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

The concept of a Framework which works over a decentralized network of Super Nodes and Nodes has never been implemented for the purpose of distributing functions. Though there are similar Systems which shares static content over a peer to peer network such as IPFS. Most decentralized systems which exists focuses on the distribution of different entities such as

1. IPFS – A decentralized static content storage platform which has the functionality identical to AWS S3.
2. SteemIt – A decentralized blogging platform based on Rewards
3. Storj – A Decentralized Files Distributing framework

Beaker Browser is decentralized network to create and host staticwebsites. It provides a user friendly interface and few easy steps to deploy a static website. Download, Create a page, Deploy are the main steps involed. Using beaker users can share personal files and make webpages. Every app is hostless and forkable source code.

The breaker browser focuses more on static content rather than dynamic content. But Orb doesn't support hosting server side code. Therefore, the concept of Orb isn't adapted anywhere in the world.

2.2. Identification and significance of the problem

The idea behind Orb is built around the concept of decentralized functions as a service. The functions reside on the nodes. With the assistance of Super Nodes, the functions are consumed by nodes spread across the network. The client nodes can deploy a function, another node may seed it and finally, the function that's deployed should be able to consumed by a SDK. To solve this, the system must overcome certain challenges and problems.

The Major research problem related to this component is that, there's no straight forward way of calling a function stored in a Node behind a NAT. Therefore, the concept behind this component is to find a mechanism to consume functions located at one private network from another private

network through a network of Super Node. Further, due to the decentralized nature of the proposed system, there's no way of reverting or removing a function once it's deployed to the network.

Therefore, as a Solution, decentralized Function Distribution Framework is proposed.

2.3. Technical objectives

The software requirements it needs the server software to be installed on a server with following requirements.

1. Dual Core CPU.
2. 4GB Ram.
3. Linux based Operating System.
4. Public IP Address.
5. Relevant ports should be open.

The Software requirements includes Orb Server software comes with the Function Distribution Framework should be installed.

2.4. Detail design

The Function Distribution Framework is implemented in Node and runs on a server with public ip address. The following figure shows the basic architecture of the Function Distribution Framework.

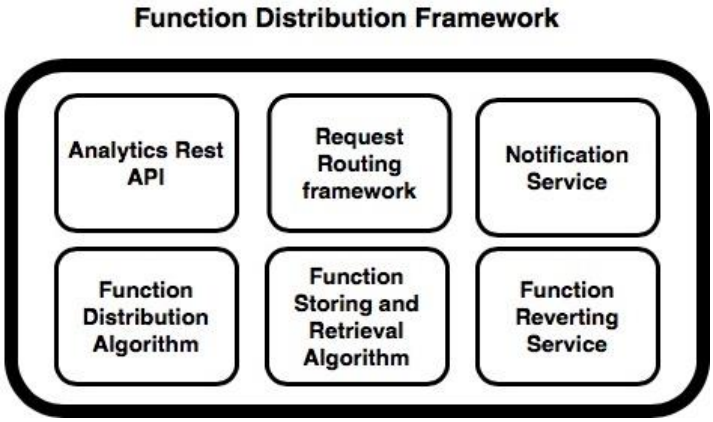

Figure 1- Architecture of the Function Distribution Framework

The Analytics REST API helps to gather important information such as

1. Server logs
2. Number of nodes connected
3. Number of super nodes connected
4. Function Manifests
5. Performance Data

This API is used for monitoring purposes and it's to be implemented using NodeJS as a lightweight micro service.

The Request Routing Framework helps the nodes to find deployed functions and route the function calls through the network, from;

1. Node to Node
2. Super Node to Super Node

The Notification service is deployed to alert the connected nodes once a function is submitted by a node. Therefore, a node could choose to seed the deployed function.

The Function Reverting Service helps to revert and rollback functions that's already deployed to the network. To revert a function, the function owner must provide the function descriptor and the client secret. Once a user makes a change to a function, the new function will replace the older function when reverting. Upon decommissioning, the request is broadcasted to the network. Slowly the request propagates through the network removing the relevant function from each seeder.

2.4.1 Resolving the issue of public and private networks

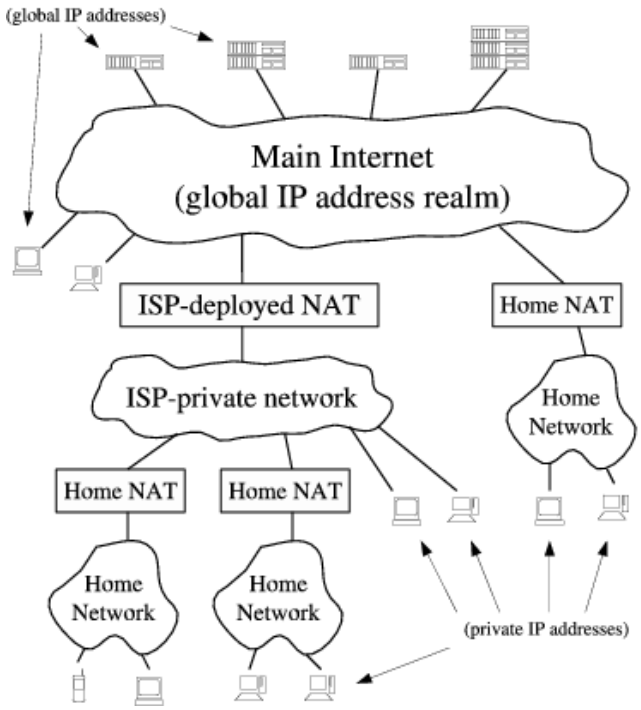

Figure 2 - Private and Public Networks

The Internet's new de facto address architecture is suitable for client/server communication in the typical case when the client is on a private network and the server is in the global address realm. The architecture makes it difficult for two nodes on different private networks to contact each other directly. Orb clearly needs a way to make such protocols function smoothly in the presence of NAT. As shown in the Figure 1, it's impossible to consume a function deployed in a node behind a NAT. To overcome this situation, a super node is used.

When the client initiates a connection to the Super Node, the Super Node passes all the available function descriptors to the Client Node. The client Node can select a function and emit to the "call-function" event on Super Node along with the Function Descriptor as Data. Then the Function Distribution Framework search its data stores for the relevant function. Then it obtains a list of nodes which hosts the function. The obtained list is sorted in ascending order based on the ping and the relevant function call is passed to the private node via the socket id. Then the relevant function with the descriptor executes inside the JavaScript interpreter in the Orb client and returns

the response back to the function caller. This way, by using a Super Node, the functionality could be achieved by overcoming the difficulties of private and public networks.

2.4.2 Function Deployments

Once a function is received by the Super Node, it will store the manifest of the function in the Servers local storage. Then the notification service will dispatch a message to all the connected nodes and super nodes. Once a node accepts to seed the function, the function as a whole is copied to the relevant node and starts the seeding process. This way the function deployment framework supports easy deployments.

2.4.3 Routing Requests

The Orb Server Software consists of the “call-function”event where all the client are subscribed. When a client wants to consume a decentralized function, the caller must emit to the “call-function” event in the Super Node through socket.io. Then the algorithm will find the relevant function and channels the request. After the request is processed by the relevant node, the response is returned to the caller.

2.4.5 Security

The security concerns pertaining to the function distribution framework are minimized by using authorization requests and using secure web sockets.

2.5. Sources for test data & analysis

Since this component is under implementation, the test data is attached as Appendix A

2.6. Anticipated benefits

2.6.1 Instant Function Distribution

Since the decentralized Function Distribution Framework is a part of the Orb network, the benefits come when this component is integrated with the Orb Network. The distribution of functions across different nodes efficiently and instantly upon the submission saves lot of time.

2.6.2 Function Reverting and Decommissioning Facility

The Function Distribution Framework offers the functionality of reverting and Decommissioning which is not offered in most Decentralized Frameworks. The IPFS offers the functionality of removing a file but it takes a longer time since IPFS shards files into smaller chunks. But since Orb doesn't store functions in different nodes as shards but as Replicas, it's easy to issue a decommissioning command which benefits the user.

2.6.3 Optimized Storage

The Function Distribution Framework Stores and transports functions as JSON thereby saving storage space in Super Nodes.

2.6.4 Efficient Function Routing

The Function Distribution Framework routes functions based on the XOR distance and the response time checked by heart beats as mentioned in the technical specification.

3 Project plan and schedule

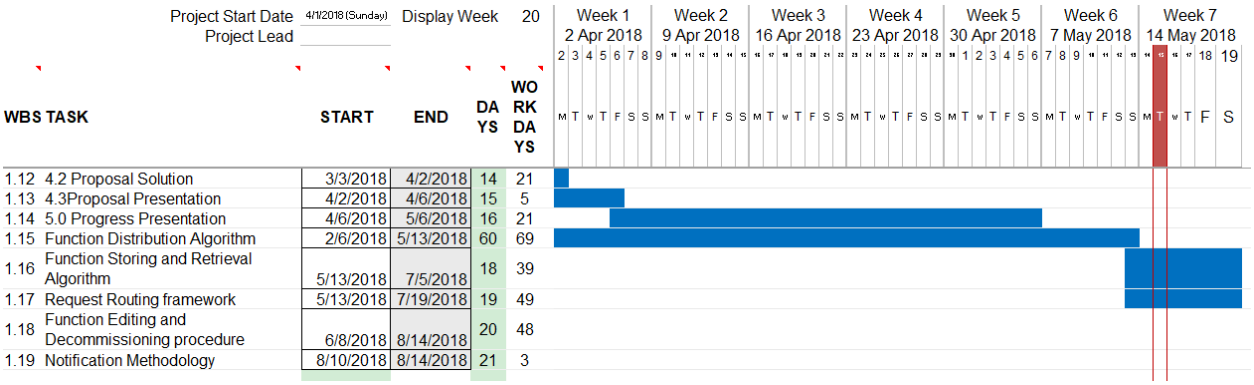

Figure 3- Gantt Chart Part 1

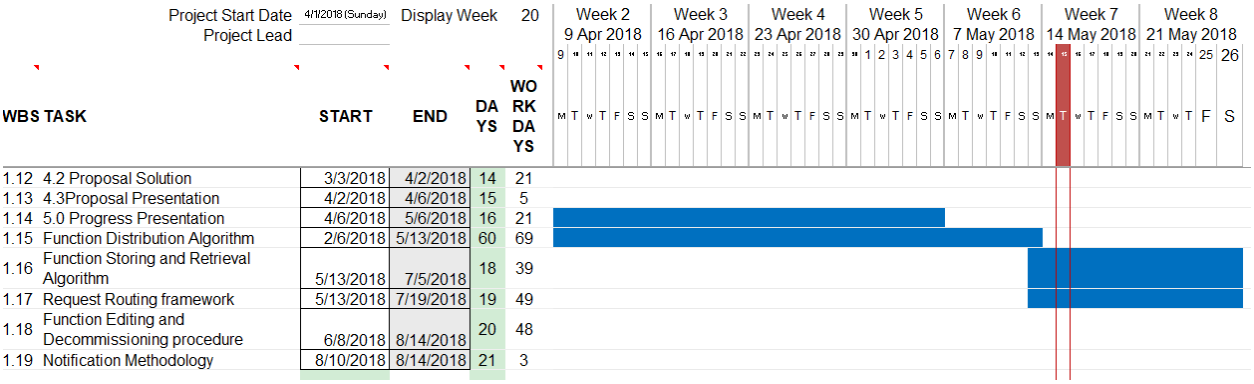

Figure 4- Gantt Chart Part 2

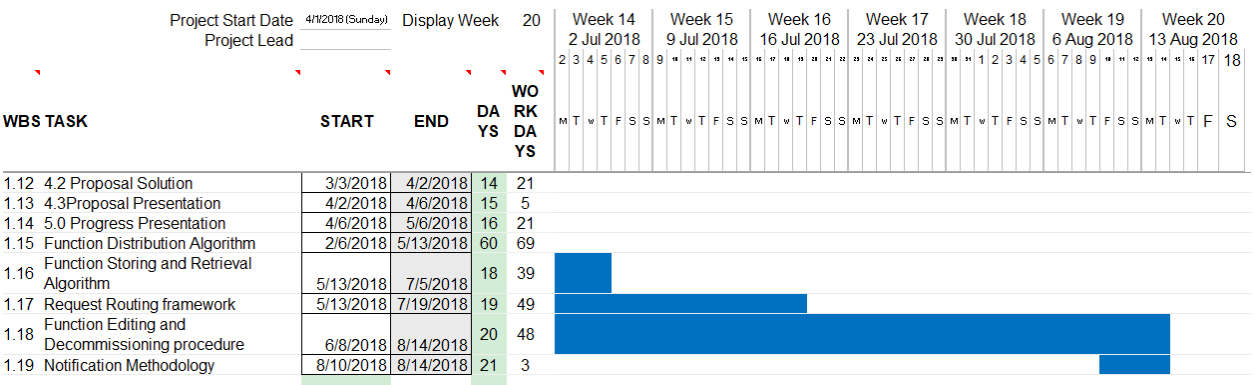

Figure 5 - Gantt Chart Part 3

4 Research constraints

4.1 Need of more than 3 Computers

The major constraint of this component is the necessity of more than 3 computers to check the functionality of the framework. This could be tested by running the Function Distribution Framework and the Client Software locally in different ports but then a client won't be able to use the deployed function though the SDK since both Super Node and the Node reside locally in a private network unless the connection is reverse tunneled to another computer with a public IP address. But then there is a configuration overhead.

4.2 Language Support

The current Function Distribution Framework only supports JavaScript. The transport language used is JSON. Therefore, in future implementations, this will be resolved.

5 Specified deliverables

The main deliverable of this research component is to deliver the Decentralized Function Distribution Framework. The Framework consists of following sub components

1. Function Distribution Algorithm
2. Function Storing and Retrieval Algorithm
3. Request Routing framework
4. Function Editing and Decommissioning procedure
5. Notification Methodology

6 Reference

[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] "pkg," npm. [Online]. Available: https://www.npmjs.com/package/pkg. [Accessed: 10-May-2018].

[28] "qtimeit," npm. [Online]. Available: https://www.npmjs.com/package/qtimeit. [Accessed: 10-May-2018].

[29] “javascript-obfuscator,” npm. Available: https://www.npmjs.com/package/javascript-obfuscator. [Accessed: 10-May-2018].

[30] "Stunnix JavaScript Obfuscator sample output", Stunnix.com, 2018. [Online]. Available: http://stunnix.com/prod/jo/sample.shtml. [Accessed: 13- May- 2018].

7 Appendices

Appendices A

7.1 Function Distribution Framework code

```
'use strict'
const express = require('express')
const ip = require('ip')
const _ = require('lodash')
const app = express()
const http = require('http').Server(app)
const io = require('socket.io')(http)
const port = process.env.port || 3000

let fnObj = []
let sendFn = []

io.on('connection', function (socket) {
    console.log('A user Connected')
    socket.broadcast.emit('user connected')
    socket.emit('event',{ msg: 'hello' })
    socket.emit('register',{ msg: 'please register' })

    socket.on('disconnect', function() {
        console.log('Got disconnect!')

        for(let fno of fnObj) {
            if(fno.socketId === socket.id) {
                fno.socketId = ''
                return
            }
        }
    });

    socket.on('reg-req', function (data) {
        const newFbObj = {
            clientId: data.clientid,
            socketId: socket.id,
            fns: []
        }

        for(let fno of fnObj) {
            if(fno.clientId === data.clientid) {
                fno.socketId = socket.id
                console.log('Fn client already exists but updated the sockets ')
                console.log(fnObj)
                return
            }
        }
```

```
    fnObj.push(newFbObj)
    console.log(fnObj)

})

socket.on('fcall', function (data) {
    const params = data.params
    const clientId = data.clientId
    const fname = data.fname

    console.log(sendFn)
    for(let fns of sendFn) {
        console.log(fns.name == fname)
        if(fns.name == fname && typeof fns.socketId != null) {
            const params = data.params
            const clientId = data.clientId
            const fname = data.fname

            let namespace = null;
            let ns = io.of(namespace || "/");
            let socket = ns.connected[fns.destinationSocketId]

            if (socket) {
                console.log("Socket Connected, sent through socket");
                socket.emit("answer-seek", { fnobj :fns, params: params },  function (answer) {
                    socket.emit('answer', answer)
                    console.log(answer)
                });
            } else {
                console.log("Socket not connected, sending through push notification");
            }

        }
    }
    console.log('----------end of function call----------')
})

socket.on('deploy',function (data) {

    const newfnObj = {
        id: data.id,
        name: data.name,
        fn: data.fn
    }

    for(let fno of fnObj) {
        if(fno.clientId == data.clientId) {
            let fnsList = _.unionBy([newfnObj], fno.fns, 'name')
            fno.fns = fnsList
            console.log(fno)
        }
    }
```

```
        for(let fnx of fnObj) {
            console.log('deploying to outside client')
            let socketId = fnx.socketId
            let clientId = fnx.clientId
            let sentObj = {
                destinationSocketId: socketId,
                destinationClientId: clientId,
                id: data.id,
                name: data.name,
                fn: data.fn
            }
            sendFn.push(sentObj)
            io.to(socketId).emit('new-fun', sentObj)
        }
    })
})


app.get('/', (req, res) => {
    res.status(200).send(JSON.stringify(fnObj))
})

app.get('/clients', (req, res) => {
    res.status(200).send(Object.keys(io.sockets.sockets))
})

http.listen(port, '0.0.0.0', e => {
    if(e) {
        console.log(e.message)
        return
    }
    console.log(`Socket server running on ${port} @ ${ip.address()}`)
})
```