# CS425 PROJECT REPORT - GROUP 13
## Decentralized Storage Network

Arham Chopra 14130 Gopichand Kotana 14249 Nikhil Vanjani 14429

## Introduction

The vision behind this project is of reorganizing the structure of cloud, because we believe there are a lot of flaws in its current structure.

Today, cloud infrastructure is centralized – for example, Google Drive, Amazon Web Serveices, etc. The centralized structure can't meet long term scaling requirements. Today, computing power and stroage capacity of computers are growing faster than bandwidth of networks. More and more data being pushed through the network is streaming data. We are moving from HD to 4k, 8k and, VR. This implies, the data being pushed into the network is growing massively, and because everything is shifting to cloud, the throughput of the cloud data centers becomes the bottleneck.

Another issue with centralization is that we are trusting these central organizations with our data. These huge private firms may not be tampering with our data but it is well known that they have to comply with nation states in providing them with data. One may debate on privacy vs security. Traditionally people trusted the nation states to use their private data not against them, but for their privacy. But, in recent years it has been well known that nation states have used such data for wrong purposes.

Last issue with centralization is that if there are security breaches in the data centers we trust, which do happen quite frequently nowadays, the whole gathered data gets leaked and this scale is massive.

## Objective

The three problems mentioned in introduction can be resolved with the emergence of decentralized and peer-to-peer (aka, distributed) technologies. Decentralization can solve second and third problem and peer-to-peer can solve the first problem. In this project, our objective is to build a decentralized storage network.

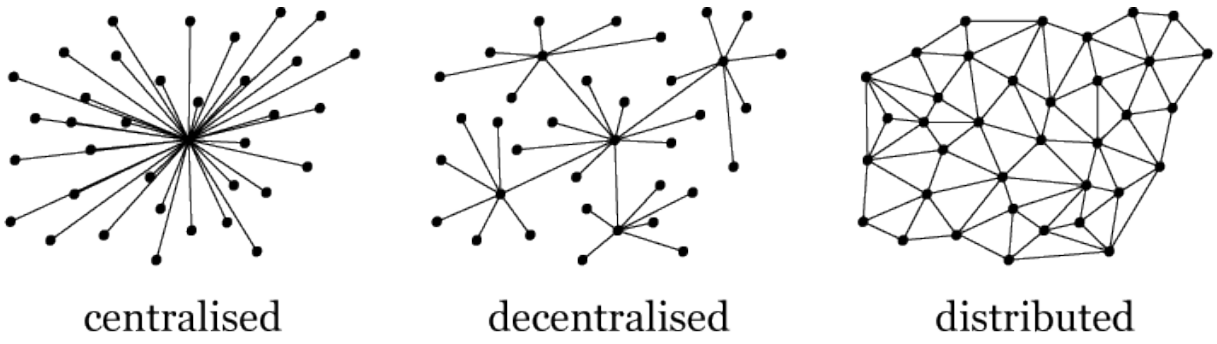



Figure 1: Source:https://commons.wikimedia.org/wiki/File%3ACentralised-decentralised-distributed.png

A decentralized network would consist of two layers of communication. First layer consists of the group of manager nodes serving the network. It usually is dynamically allocated and not static for security reasons. The second layer consists of each of these manager nodes interacting with client nodes connected to them and serving their queries. Because the duration for this project was short, we implement only the second layer of such a network. Once this layer is in place, we could build the first layer on top of it.

## Assumptions

- The number of copies of a file is kept constant when the file is being uploaded for the first time. We rely on the fact that with high probability, at all times, at least one of the nodes storing each file would be active and can serve the file. Ideally, when the number of copies goes below a certain threshold, more copies should be made to ensure availability at all times.
- A drawback of our network is that it is possible that a client wants to upload a huge file, the space for which is may not be available on any of the storage nodes. In such cases, upload will fail.

## Architecture

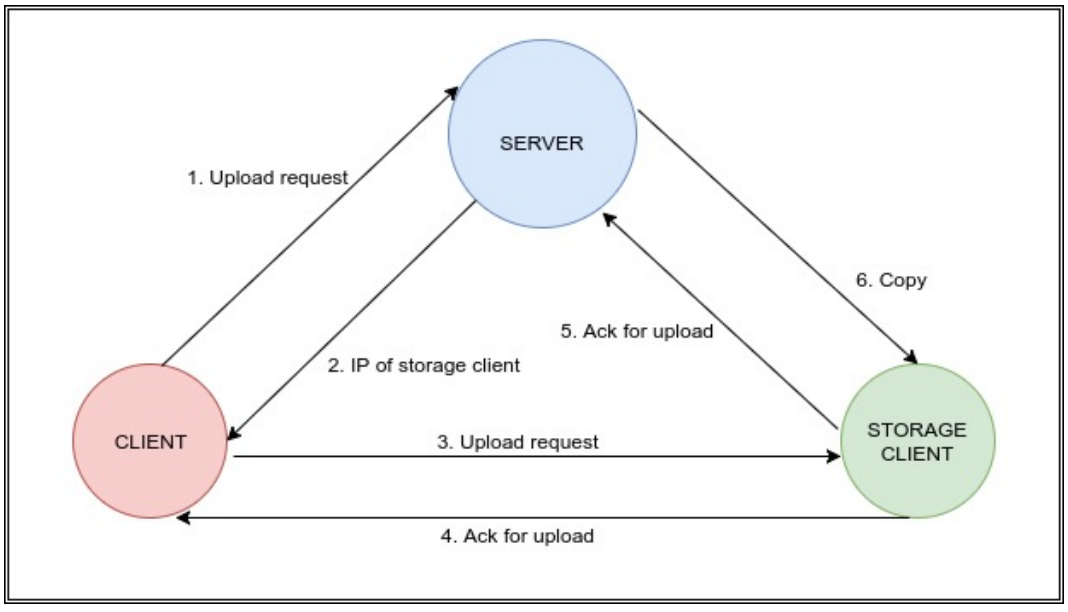
*There are three kinds of nodes on the network*

- Client
- Storage Server, essentially some clients themselves
- Central Server

*There are three protocols which make the network a reliable data storage medium: Upload, Download, Copy*

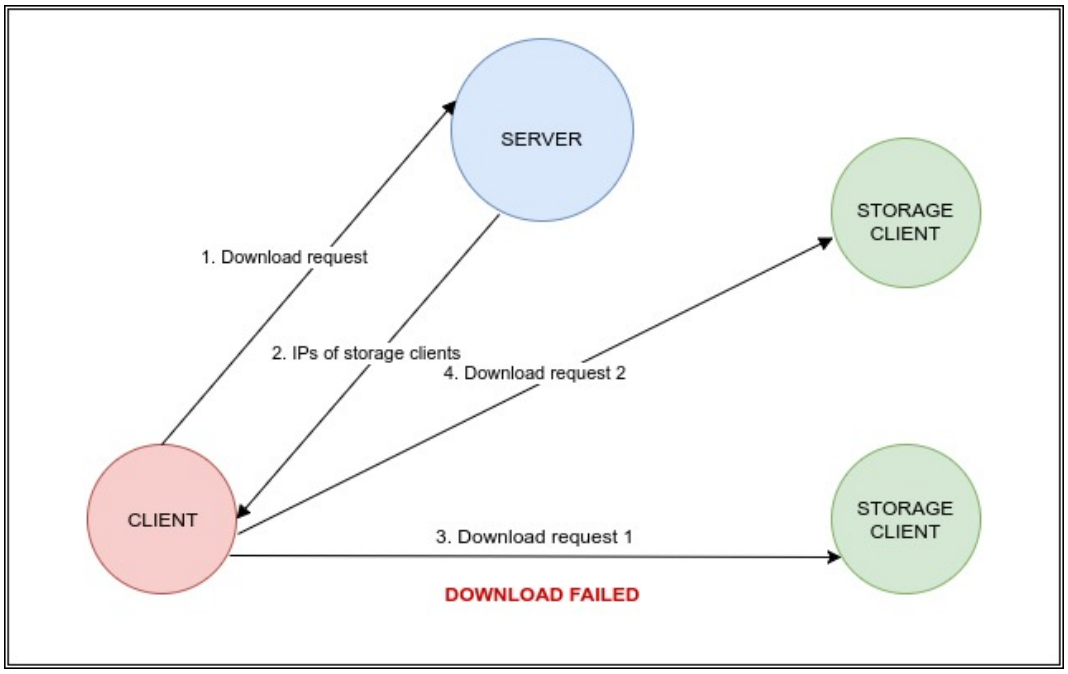
### Upload

To upload a client's file to one of the storage client; The server provides an IP address to the client upon receiving a request for an upload. The client uploads the file to the machine with the particular IP. Then the server takes over and makes copies of the files to allow for storage client crashes, by using the copy protocol.



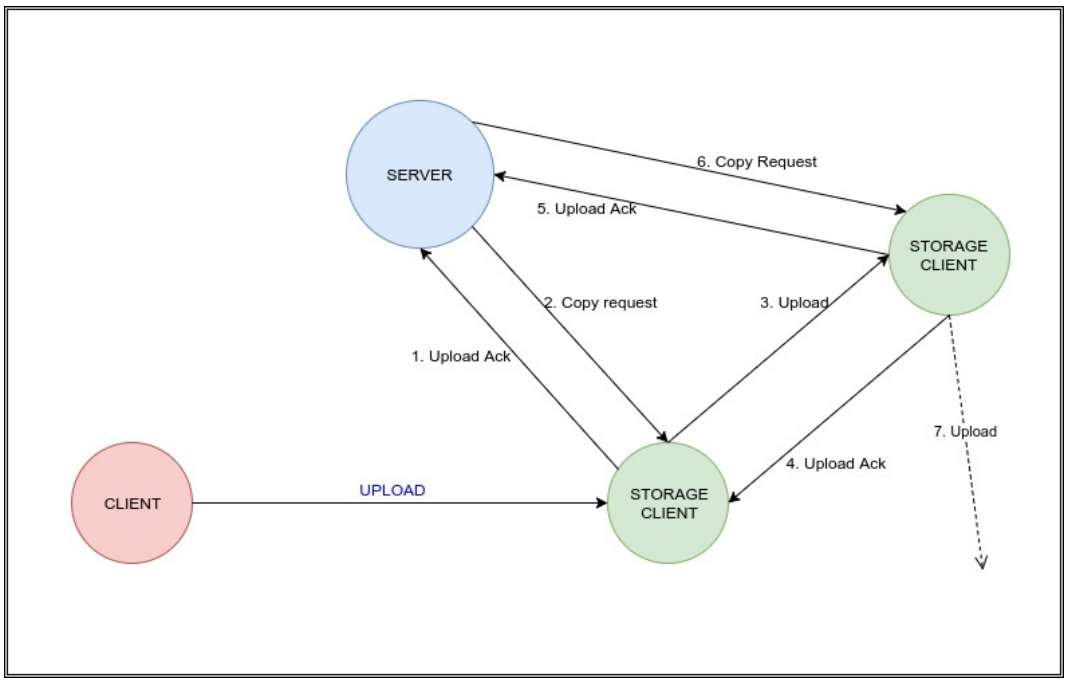
### Download

To download a client's file from a storage; The client sends a download request to the server which provides it with a list of IPs of machines which have the particular file. The client requests the storage client(selected from the list) for the file. The client iterates over the received list of IPs until it receives the file it wants to download.




### Copy

To copy a file to multiple storage clients, to ensure the file remains available for download even when a storage client goes down; After a client finishes an upload to a storage client, the server requests the storage client to copy the file to another storage client. This is done multiple times by the server to copy the file to several nodes.



### Database Architecture

**Tables:**

- FileData
  - FileId
  - FileName
  - Owner
  - IP_list
  - Size
- StorageData
  - StorageId
  - StorageIP
  - StorageSpace
  - UsedSpace
  - Status
  - FileLock

## Implementation environment

- Linux distro
- Python2.7
  - Libraries : MySQLdb, argparse, socket, atexit, os, random, time, threading, subprocess
- MySQLdb
- nmap

## Summary

The initial proposal was to implement the following protocols: upload, download, copy, add storage, remove storage, remove file and implement the respective database queries. We have implemented, upload, download, copy and, add storage protocols. We haven't implemented remove storage and remove file protocols.

Lastly, we should mention that our project was inspired primarily from MaidSafe Network, which is a distributed and decentralized network for storage and computation. It also implements its own cryptocurrency for managing transactions and bootstrapping the network.

The repository of the code is currently closed as the deadline to submit code has not passed. Please send us your bitbucket id so that we can give you access to the repository.