

BLOCKCHAIN INCENTIVIZED AND SECURED PEER TO PEER FILE SHARING PROTOCOL

PROJECT REPORT

SUBMITTED IN PARTIAL FULFILLMENT OF THE
REQUIREMENT FOR THE AWARD OF THE DEGREE OF

BACHELOR OF TECHNOLOGY

IN

COMPUTER ENGINEERING

SUBMITTED BY

AYUSH GUPTA **2K15/CO/040**

DEVANSHU KHURMA **2K15/CO/052**

HARSHIT AGGARWAL **2K15/CO/059**

UNDER THE SUPERVISION OF

PROF. RAJNI JINDAL



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

DELHI TECHNOLOGICAL UNIVERSITY

(FORMERLY DELHI COLLEGE OF ENGINEERING)

Bawana Road, Delhi-110042

Department of Computer Science and Engineering

DELHI TECHNOLOGICAL UNIVERSITY

(Formerly Delhi College of Engineering)

Bawana Road, Delhi-110042

CANDIDATE'S DECLARATION

We, **Ayush Gupta (2K15/CO/040)**, **Devanshu Khurma (2K15/CO/052)** and **Harshit Aggarwal (2K15/CO/059)** of B. Tech. (Computer Engineering), hereby declare that the project dissertation titled “**Blockchain Secured and Incentivized Peer to Peer File Transfer Protocol**” which is submitted by us to the Department of Computer Science and Engineering, Delhi Technological University, Delhi in partial fulfillment of the requirement for the award of the degree of Bachelor of Technology, is original and not copied from any source without proper citation. This work has not previously formed the basis for the forward of any Degree, Diploma Associate-ship, Fellowship or other similar title or recognition.

Place: DTU, Delhi	AyushGupta	Devanshu Khurma	Harshit Aggarwal
Date: 28/05/2019	2K15/CO/40	2K15/CO/052	2K15/CO/059

Department of Computer Science and Engineering

DELHI TECHNOLOGICAL UNIVERSITY

(Formerly Delhi College of Engineering)

Bawana Road, Delhi-110042

CERTIFICATE

I hereby certify that the Project Dissertation titled “**Blockchain Secured and Incentivized Peer to Peer File Transfer Protocol**” which is submitted by **Ayush Gupta (2K15/CO/040)**, **Devanshu Khurma (2K15/CO/052)** and **Harshit Aggarwal (2K15/CO/059)** Department of Computer Science and Engineering, Delhi Technological University, Delhi in partial fulfillment of the requirement for the award of the degree of Bachelor of Technology, is a record of the project work carried out by the students under my supervision. To the best of my knowledge this work has not been submitted in part or full for any Degree or Diploma to this University or elsewhere.

Place: DTU, Delhi

Date: 28th May 2019

Prof. Rajni Jindal
Head of Department
Computer Science and Engineering
Delhi Technological University

ABSTRACT

P2P file sharing has revolutionized how we deal with digital content in recent years. With abundance of seed boxes and torrent web sites galore, P2P data sharing is emerging as the preferred method of digital data transfer and sharing. Primarily because of the numerous advantages it offers over conventional server to client model such as downloading speed and content availability to name a few. At the same time this way of sharing digital data suffers from some inherent flaws that we wish to tackle in our work.

First and foremost, security is a big point of concern in P2P sharing since they are open for anyone to participate in. Corrupted data could be distributed on a P2P network by modifying files that are already being shared on the network. We will be dealing with the spread of malicious files by tracking the origin of every data packet. Every node acts as a client as well as a server due to which it is very vulnerable to remote exploits. Each node plays a role in routing packets through the network causing it to be susceptible to routing attacks by malicious users for example a DDoS attack.

Another significant problem arises from the reluctance of most ‘leechers’ to ‘seed’ after they are finished downloading the content. This is a particularly tricky challenge since the status quo way of leaving the responsibility to seed at least as much data as they download pro bono is proven to be ineffective whereas making it mandatory to do so will drive potential users away to less follow-up action binding alternatives. The best solution therefore seems to be the middle ground of letting the users decide how much they want to seed but at the same time incentivize their seeding to promote this behavior without driving them away using a cryptocurrency. The confluence of P2P data transfer and Blockchain technology carries immense potential that we aim to utilize.

ACKNOWLEDGEMENT

“The successful completion of any task would be incomplete without accomplishing the people who made it all possible and whose constant guidance and encouragement secured us the success.”

First of all we are grateful to **Prof. Rajni Jindal, HOD (Department of Computer Science and Engineering)**, Delhi Technological University (Formerly Delhi College of Engineering), New Delhi and all other faculty members of our department, for their astute guidance, constant encouragement and sincere support for this project work.

We owe a debt of gratitude to our guide, **Prof. Rajni Jindal, HOD (Department of Computer Science and Engineering)** for incorporating in us the idea of a creative Project, helping us in undertaking this project and also for being there whenever we needed his assistance.

CONTENTS

Abstract

Acknowledgement

Certificate

1. Introduction

1.1 Objective of our project is five-fold:

1.2 Our Proposed outcomes are listed below:

2. Review of Literature

2.1 Peer to Peer File Sharing

2.1.1 The oracle problem and why we are safe

2.1.2 The Free Riding Problem

2.2 Bit Torrent

2.2.1 Introduction

2.2.2 Operation

2.2.3 Creating and Publishing Torrents

2.2.4 Downloading Torrents and Sharing of Files

2.3 Blockchain

2.3.1 Fundamental traits of Blockchain technology

2.3.2 Structure and characteristics of Blockchain

2.4 Crypto Currency

2.4.1 Ethereum

2.4.2 Bitcoin

2.5 Smart Contracts

2.6 InterPlanetary File System (IPFS)

2.6.1 Overview

2.6.2 Distributed Hash Tables

2.6.3 Merkle DAG

2.7 Elastic Search

2.8 Logstash

2.8.1 Introduction

- 2.8.2 The Power of Logstash
- 2.9 Kibana
- 3. Methodology
 - 3.1 System Architecture
 - 3.1.1 Currency Distribution Scheme
 - 3.1.2 Maintaining Security of Files
 - 3.1.3 Maintaining Integrity of Files
 - 3.1.4 Searching for Content on Client Side
 - 3.1.5 Managing Uploading on Client Side
 - 3.1.6 Creation of New File
 - 3.2 Smart Contract Architecture
 - 3.2.1 Files Base
 - 3.2.2 Users Base
 - 3.3 Development Environment and Tools
 - 3.3.1 Development Tools
 - 3.3.2 Languages/Frameworks Used
 - 3.4 Data Flow Diagram
 - 3.5 Contract Code Snippets
- 4. Report on Present Investigation
- 5. Case Study
 - 5.1 Running a Network Simulation
 - 5.1.1 Popularity Scores
 - 5.1.2 Parameters
 - 5.1.3 Data Summary
 - 5.1.4 Test Values:
 - 5.1.5 Observations
- 6. Results
 - 6.1 SteemIt Versus Our Platform
 - 6.2 Future Work and Improvements
- 7. References

List of Figures

Plot: cost of downloading Vs Popularity score	Page 29
Phase 1: Uploading	Page 35
Phase 2: Downloading	Page 36
Phase 3: Incentivized Seeding	Page 36
Figure 5: Searching the Filebase	Page 39
Figure 6: Elastic Search in Action	Page 40
Figure 7: Choosing to Download	Page 41
Figure 8: Paying the cost of Download	Page 42
Figure 9: Download begins as soon as transaction is validated	Page 42
Figure 10: Download of torrent file is done	Page 43
Figure 11: Transaction to pay cost of download is verifiable	Page 43
Figure 12: The transaction as visible to everyone	Page 44

CHAPTER 1 INTRODUCTION

P2P file sharing has revolutionized how we deal with digital content in recent years. With abundance of seed boxes and torrent web sites galore, P2P data sharing is emerging as the preferred method of digital data transfer and sharing. Primarily because of the numerous advantages it offers over conventional server to client model such as downloading speed and content availability to name a few.

At the same time this way of sharing digital data suffers from some inherent flaws that we wish to tackle in our work.

First and foremost, security is a big point of concern in P2P sharing since they are open for anyone to participate in. Corrupted data could be distributed on a P2P network in the guise of useful program files that are readily downloaded and further spread by gullible users on the network.

We will be dealing with the spread of malicious files by verifying the nature of files being attempted to be shared by first running them and inspecting their behavior in a virtual environment. This is implemented through a quorum-based voting algorithm where the quorum members are also incentivized for their services.

Every node acts as a client as well as a server due to which it is very vulnerable to remote exploits. Each node plays a role in routing packets through the network causing it to be susceptible to routing attacks by malicious users for example a DDoS attack.

Another significant problem arises from the reluctance of most ‘leechers’ to ‘seed’ after they are finished downloading the content. This is a particularly tricky challenge since the status quo way of leaving the responsibility to seed at least as much data as they download pro bono is proven to be ineffective whereas making it mandatory to do so will drive potential users away to less follow-up action binding alternatives.

The best solution therefore seems to be the middle ground of letting the users decide how much they want to seed but at the same time incentivize their seeding to promote this behavior without driving them away using a cryptocurrency.

The confluence of P2P data transfer and Blockchain technology carries immense potential that we aim to utilize.

1.1 Objective of our project is five-fold:

1. Making P2P file transfers more secure and trustable
2. Dealing malware injection in the network and restricting its propagation
3. Identifying seeder and leecher nodes in the network
4. Incentivizing seeding on the basis of above analysis
5. Comparative analysis security level with the current state-of-the-art BitTorrent protocol

To safeguard against the conventional P2P protocol's drawbacks, we will get a fixed number of members to form a quorum. This quorum is asked to judge the safety of the file to be shared before it is made available to all users. A simple vote-taking is done and if the majority vote deems the file "safe for propagation", then only is this file allowed to be shared in our protocol.

The seeders will be incentivized for their positive contribution to the network. Access to the download will require a user to pay a certain price in a cryptocurrency and this amount will be refunded as the user seeds data after finishing the download thereby maintains a healthy availability of the file.

1.2 Our Proposed outcomes are listed below:

1. Enhanced security of P2P file transfer
2. Promotion of seeding behaviour in leechers through an incentivization layer
3. Comparative analysis between existing P2P protocols and our work
4. Achieving availability proportional to the utility of data and not just popularity of data

CHAPTER 2 REVIEW OF LITERATURE

REVIEW OF LITERATURE

2.1 Peer to Peer File Sharing

P2P File Sharing Protocols originated in late 1990s and have been getting increasingly popular and convenient to use ever since. P2P network allows two systems to communicate with each other directly thereby eliminating the need of a server to obtain a digital file by searching and sourcing them from other end-user computers that act as nodes connected on the P2P network. This marked the beginning of shift towards a decentralized paradigm which was further promoted by increasing bandwidth of the internet, the dawn of digitization of various types of media ranging from books to photographs to music records, and increasing capabilities of end-user computers through more powerful, compact, and faster CPUs. The most widely used and arguably the most successful application of P2P protocols is BitTorrent.

Cost Efficiency of P2P networks is another good reason for their popularity. Each network can be monitored by users themselves since the provider is usually also the administrator thereby reducing the system administration overhead. Also, because large servers require more storage and this leads to added costs since this storage is purchased to be used only for the use by the server and this need for dedicated server is also eliminated in a P2P network.

Peer-to-Peer file sharing is a way to legitimately share the with distribution rights like Creative Commons, Copyleft. Even the media that is no longer copyright protected and openly available for use in the public domain.

Start-ups are particularly benefitted from using P2P networks rather than having to use other Content Delivery Modes

With peer to peer file sharing:

$$Total\ Cost = File \frac{Size}{Customers} \times Cost\ per\ Byte$$

Whereas with casual content delivery systems:

$$Total\ Cost = File \times Customers \times Cost\ per\ Byte$$

P2P networks suffer some inherent flaws as listed below: -

Several modern-day companies work along similar lines: finding suitable customer and service provider matches using algorithms. Food delivery startups like Zomato, UberEats match delivery guys with restaurants and customers. Ride-hailing apps like Uber matches drivers with passengers. Cryptocurrency, as well as fiat exchanges, match shareholders interested in selling with those interested in buying. eBay matches buyers with sellers. Dating and marriage apps like Tinder, shaadi.com matches possible suitors with each other. It is surprising to know that this modern phenomenon has a term on its own: Uberization. All these companies have one thing in common—they are managed and run by central authorities who are the ones making all the administrative and management decisions. They have the final word in case of any dispute. With the help of blockchain, people are attempting to convert these seemingly peer-to-peer but actually centralized networks to fully decentralized ones.

All such attempts get stuck on one major problem, the Oracle Problem.

2.1.1 The oracle problem and why we are safe

The oracle provides data about the real world to the smart contract. For example, in the case of Uber, it will tell the contract about taxi details. In Zomato, it will record data about food items[State example properly. The problem is if the oracle is compromised, so is the entire contract.

The Augur team has spent years towards working towards a solution, but it still isn't perfect.

This problem, which is currently unsolved according to many people, does not seem to affect our system. This is because of the simple reason that we are not dealing with any external, tangible entity. The files are uniquely identified using the content addressable hash] generated when the user adds it to the system for the first time. The users are identified by their wallet address. Thus we do not have to trust any third party or human in the process.

2.1.2 The Free Riding Problem

This is a particularly tricky challenge since the status quo way of leaving the responsibility to seed at least as much data as the user download pro bono is proven to be ineffective whereas making it mandatory to do so will drive him away to less follow-up action binding alternatives.

2.2 Bit Torrent

2.2.1 Introduction

It originated among the first generation of filesharing protocols that relied on centralized servers in the role of tracker nodes that were responsible for coordinating the users but it does not form a network in the traditional sense since every new set of files gets its own network of coordinating users called torrent.

The advent of DHT and its application called mainline DHT rendered the need for centralized trackers obsolete. Mainline DHT is a decentralized server-independent network for source identification purposes.

Users upload the index files that contains information about the metadata of the files they wish to share and upload them on websites where other users can find them. This is a typical feature of second-generation filesharing networks.

The key differences between BitTorrent protocol and the classic download through HTTP and FTP requests are listed as follows: -

- BitTorrent generates numerous requests for small amounts of data through dissimilar Internet Protocol connections to various machines, whereas traditional way to download is characteristically using a single Transmission Control Protocol connection to a single machine.
- BitTorrent does downloading in an arbitrary or in a "rarest-first" style to ensures very high availability and non-sequential data download, while classic downloads are sequential.

It is owing to these points of difference that BitTorrent burdens the content provider with a significantly less cost, significantly more redundancy and far more resistant to abuse than regular

server software. The only tradeoff being that it may take some time before the download speed becomes fast which happens only after enough peer-to-peer connections have been made and it might take a peer in the network to receive enough data before it can act as an efficient active uploader. The conventional HTTP server outperforms BitTorrent in this regard since the download speed rises to maximum quickly and is maintained throughout although at the cost of being susceptible to abuse and overload through DoS attacks for example. Random order in which the file components are downloaded also keeps the user from being able to stream media files as they are downloading. In 2013 However a BitTorrent Streaming protocol's beta version was released

2.2.2 Operation

A program which implements the BitTorrent Protocol is known as a BitTorrent client. Each such program has the ability to prepare, transmit and distribute any kind of digital file over the torrent network through the BitTorrent protocol.

A Peer refers to any node running the instance of the BitTorrent client.

In order to share a group or even just a single file, a peer needs to create a file known as a torrent. This torrent file holds the metadata of the digital files to be shared and also regarding the tracker which is the node that regulates and manages the dissemination of the file.

Pers who wish to download a digital file must first acquire a torrent file and establish connection to the tracker specified in the torrent file that dictates them about downloading the fragments of the file through which other members or peers.

2.2.3 Creating Torrents

The file is treated as a number of similarly sized pieces, commonly with sizes as a 2's power, and characteristically each is between 32 kilobytes to 16 megabytes. For each piece a hash is created through SHA-1 hash function and stores it in torrent file's metadata. Fragment that have sizes over 512 kilobytes reduce the torrent file's size for a significantly bulky payload, but it is claimed that it reduces the torrent file's size for very bulky payload but this will negatively impact the protocol's efficiency. Whenever another user receives a fragment of that file, the hash of this fragment is matched with the hash in torrent file to ensure that it is free from error.

Peers that are capable of providing complete files are called **Seeders** and the peers that are providing the file for the first time are called initial seeder.

Different versions of the BitTorrent protocol have minor differences in how and what information is stored in the .torrent files.

Torrent files have an ‘announce section’ that mentions tracker’s URL and it also has an info section that possesses suggested names of the files along with their lengths, their piece-length that is used and a SHA-1 hash code of each fragment. These all are used to ensure the integrity of the received pieces as received by the leecher. BitTorrent protocol is now about to move to SHA-256.

The .torrent files are usually listed on particular websites and are registered with at the very least, 1 tracker that maintains the clients’ list that are presently participating in the swarm of the torrent. Otherwise, in a tracker-less system that is also known as decentralised tracking wherein every peer plays the role of a tracker. BitTorrent implements decentralised tracking through a Distributed Hash Table method.

Presently, most of the popular BitTorrent clients use Mainline DHT system that is an alternate and incompatible system.

Following the widespread adoption of DHT by BitTorrent clients, a “private” flag, similar to broadcast flag – was unofficially introduced, telling clients to restrict the use of decentralised tracking regardless of the user’s desires. The flag is deliberately placed in the info section of the torrent so it can not be disabled or removed without changing the identity of the torrent.

The purpose of this flag is to prevent downloading by clients that do not have access to the tracker. Clients banning the flag are blocked by many trackers thereby encouraging its adoption.

2.2.4 Downloading Torrents and Sharing of Files

Users search the .torrent file of their use listed on websites or other means and this file is then opened using the BitTorrent Client. The client then establishes the connection to trackers specified in the torrent file which is from where it derives a peer’s list that are currently disseminating pieces of the files in the torrent. The BitTorrent client obtains the several fragments of the required files after connecting to those peers. If swarm is comprised only of the initial seeder then user directly starts requesting file from it.

Various mechanisms are implemented by the clients to optimise upload and download rates. For example randomised fragment downloading to enhance the opportunities for the exchange of data that is only done when the two peers possess different fragments of the file.

The data exchange's efficiency and effectiveness largely rely on those policies that the clients make use of to decide on to whom should they send data next. Clients might prefer to transfer data to users that transfer data to them, this is called the tit-for-tat scheme. This does promote fair trading but this might lead to sub-optimal situations wherein users that have newly-joined are not able to get any peer to send them data because they do not yet have any fragments to trade against or when two peers with fragments required by each other do not communicate because no peer takes the initiative. To deal with this problem most client programs implement 'optimistic-unchoking' wherein a client saves a part of the bandwidth available to it for transmitting pieces to peers arbitrarily or at random that are not decidedly beneficial partners but are known as 'preferred' peers in anticipation of discovering better and more beneficial partners and to make sure that newly-joined peers get an opportunity to join the swarm.

Swarming can be scaled well to cope with "flash crowds" which are commonly observed for a popular content, it is of less utility for less popular or niche market content. The members of swarm attempting to download once the initial rush is over may have to face the content's unavailability and the compulsion to wait until the seeders start joining the swarm and being active to complete their downloads. The seeder's arrival may happen after extended period of time until which there is no active seeder in the swarm and this is referred to as the "**seeder promotion problem**".

Because maintaining active seeders of niche and not as popular content exerts high administrative and bandwidth cost, it seems to be against the driving idea of BitTorrent protocol to be a cheaper alternative option to a client server model and this is observed on a good scale.

A study claims that as much as 38% of torrents become unavailable within their first month after listing. Many initial seeders circumvent this problem by propagating the files containing such niche data with other more widely known and popular files in just a single swarm. More elaborate solutions are also suggested that rely on better cross torrent communication to better cooperate at content sharing.

2.3 Blockchain

2.3.1 Fundamental traits of Blockchain technology

- **Decentralization:**

Centralized services relying on server software and HTTP requests to a single server in a system that generally speaking consists of an entity that is centralized and the client needs to interact with this entity alone to get the required information. These centralized systems have been the norm until very recently where BitTorrent and Bitcoin have been gaining widespread popularity.

Take the example of banks which keep all their client's money stored and the only way a client can pay someone is through the bank. Whereas in a decentralized network, if one wants to pay someone else, he or she could do that directly thereby completely eliminating the need for a third party, this was also the main idea behind Bitcoin. Only the person paying is in charge of his money and should be able to send it to whoever they want without having to go through a bank.

- **Transparency:**

Often misunderstood by a lot of people who often get confused between the privacy of blockchain and its seemingly contradictory transparency that it promises.

In practice, the users' identity is hidden using cryptographic techniques such as hashing. Looking at a person's transaction history will not reveal that he sent a specific amount of cryptocurrency to another particular user, instead it will be in the form of a hash generated by standard cryptographic algorithms like SHA-256 in the form of say, "J3J4JN9SKSEF3j2 sent 2 BTC".

So, even though a person's real identity is hidden, one can still see and verify all the transactions made by his public address and the transparency and openness has never existed in a financial system before and provides that level of accountability that is required by the

biggest institutions. So if one knows the public address of any big corporation even, one can simply use an explorer to scrutinize all the transactions that they have engaged in which forces everyone to be honest which is a novel restriction that was not properly enforced in centralized systems. But the corporations will realistically speaking, never make transactions in cryptocurrencies but if blockchain was to be an inherent part of their functioning as in their supply chain, for example then they are compelled to be more accountable than ever before.

- **Immutability:**

In the domain of blockchain, immutability refers to the persistence of something that has been entered into the blockchain. It means that anything stored on the blockchain can not be tampered with unless the attacker controls 51% of the computing power of all the systems maintaining consensus on the state of blockchain which is an infeasible task in terms of energy and money requirements to undertake. The most glaring example of utilizing this trait is to prevent a company against embezzlement incidences since a potential embezzler can no longer manipulate the accounts books to get away with stolen money. Even huge amounts of data could be easily verified by matching the hash of a data to hash of the data known to be untampered and accurate provided, the hash is generated using the same hashing algorithm that output the hash of a pre-defined fixed length irrespective of how voluminous the input was. Owing to the property called avalanche effect of hash function, even a slight change in the data massively changes the hash output.

2.3.2 Structure and characteristics of Blockchain

A blockchain can be thought of as a distributed, decentralised and public digital-ledger that is used to store transactions spanning a vast multitude of computers so the data can be altered retroactively without altering the subsequent blocks of the blockchain. This enables the contributors to check the integrity of transactions and review transactions autonomously and at a significantly low expense. Blockchain databases are managed independently through a Peer-To-Peer network and a decentralised server for timestamping. These are validated by mass co-operation fuelled by mutual self-interests. A design like this enables failure resistant work sequence and the participants' apprehension regarding security of data is negligible. Using a blockchain eradicates

the typical feature of unlimited ability to make a copy of itself from an asset which is digital in nature. It reinforces that every unit of value was sent only once, thereby alleviating satisfactorily the long-standing concern of 'double-spending'. Blockchain is sometimes defined as a *value-exchange protocol*.-A blockchain can maintain rights to a title since, when it is set up properly to explicitly describe the terms of the exchange agreement, it acts as a record that makes offer and acceptance mandatory.

- **Blocks**

Blocks store groups of authenticated transactions that are passed through a hashing algorithm and the result of which is stored and encoded into a Merkle tree. Every block stores the cryptographic hash obtained from the previous block in the blockchain which in a way links the both of them. The linked blocks can be thought of as forming a chain. This iterative method ensures the integrity of the prior block, this goes on the entire way to the first genesis block.

At times, separate blocks can be generated concurrently, leading to a condition known as a temporary fork. Along with a secure history based on hash codes, every blockchain has a particular logic for assigning scores to various versions of history so as to facilitate the one with the highest value to be selected after a contest with others. Blocks which not chosen for inclusion are termed orphan blocks.

Members verifying the database can have varying editions of the chronicle at times. They store just the highest-scoring edition of the database they know of. Whenever a member gets a higher-scoring edition (generally the older version with one new block appended) they update their database and re-transmit the updation to their connected fellow members. Although 100% guarantee of any specific transaction will persist in the highest-scoring edition of the history always cannot be given. Blockchains are characteristically designed to sum the score of newer blocks along with old blocks and offered incentives to expand with new blocks instead of overwriting old blocks. Which is why, the chance of a transaction getting superseded diminishes exponentially as more blocks are mined and added on top of it, and soon becomes negligibly small.

The block time refers to the average time taken to generate one new block and adding it to the blockchain. This can be as small as five seconds. A shorter block time enables quicker transactions. For Ethereum it is from 14s and 15s and for bitcoin it is 10 minutes.

- **Decentralization**

Storing the data on its peer-to-peer network is how the blockchain removes numerous risks which are inherent flaws in a protocol with data being held centrally. The decentralized blockchain may use ad-hoc message passing and distributed networking.

Peer-to-peer blockchain networks do not have singular points of failure that for attackers to abuse. The usage public-key cryptography is also prevalent for security.

All peers of the decentralized system store a replica of the blockchain. Quality of the data is sustained by large scale replication of databases. Any user is trusted the same as any other user and there is no concept of true or original copy. All transactions are broadcasted to all the members.

Validation of transactions is done by what are known as mining nodes that add the transaction to the new block being built by them which is subsequently added to the blockchain and this updation is then broadcasted to other members

- **Arguments for and Against Openness**

They despite of being open for all to see need physical access and are deemed more user friendly. Since all earlier blockchains were permission-less, a debate is ongoing regarding whether or not the private blockchains qualify to be known as blockchain since in a private system with verifiers assigned the work and are given permission by a centralised authoritative figure.

Supporters of the motion of above mentioned debate advocate that the terminology of "blockchain" can be applied to every data structure that groups data into time-stamped chunks or blocks. These blockchains act as a distributed variant of multi version concurrency control (MVCC) in databases. The way MVCC prohibits more than one

transactions to simultaneously modify a single entry in the database, blockchains keep more than one transactions from expending the same singular output.

Opponents claim insists however, that permissioned systems are like conventional corporate databases, that neither support decentralised data verification and nor are they made robust to withstand operator tampering or revision.

- **Demerits of a private blockchain**

It has been suggested that There is also no concept like that of a '51 percent' attack with respect to a private blockchain, since they are already controlled a 100 percent by central authority. Only having access to tools used for blockchain creation on a private server owned by a corporate, amounts to control of their entire network and change transactions in any way. This gives a disproportionate amount of authority to a selected few users and causes a lot of harm in situations of crisis.

Decentralised blockchains like the bitcoin blockchain are kept secured and consistent due to the huge effort of massive collaborative mining. Whereas in case of private blockchain, there is sheer lack of incentive or a person to mine blocks faster than their competitor effectively rendering the system as an obtusely complicated database devoid of all the features which are desirable from a blockchain.

2.4 Crypto Currency

As indicated by Jan Lansky, a cryptographic money is a framework that meets six conditions:

1. The framework does not require a focal expert, its state is kept up through dispersed agreement.
2. The framework keeps an outline of digital currency units and their proprietorship.
3. The framework characterizes whether new digital money units can be made. On the off chance that new digital money units can be made, the framework characterizes the conditions of their birthplace and how to decide the responsibility for new units.
4. Ownership of digital money units can be demonstrated solely cryptographically.

5. The framework enables exchanges to be performed in which responsibility for cryptographic units is changed. An exchange articulation must be issued by an element demonstrating the present responsibility for units.
6. If two unique directions for changing the responsibility for same cryptographic units are all the while entered, the framework performs all things considered one of them.
7. A cryptographic money (or digital money) is an advanced resource intended to fill in as a mechanism of trade that utilizes solid cryptography to verify budgetary exchanges, control the making of extra units, and check the exchange of benefits. Cryptographic forms of money utilize decentralized control instead of unified advanced cash and focal financial frameworks.
8. The decentralized control of every digital currency works through appropriated record innovation, ordinarily a blockchain, that fills in as an open monetary exchange database.
9. Bitcoin, first discharged as open-source programming in 2009, is commonly viewed as the principal decentralized digital money. Since the arrival of bitcoin, more than 4,000 altcoins(alternative variations of bitcoin, or different cryptographic forms of money) have been made.

2.4.1 Ethereum

Ethereum is an open programming stage dependent on blockchain innovation that empowers designers to fabricate and send decentralized applications.

Ethereum is a decentralized stage that runs smart contracts: applications that run precisely as modified with no plausibility of personal time, oversight, misrepresentation or outsider impedance.

These applications keep running on a custom fabricated blockchain, an immensely amazing shared worldwide foundation that can move an incentive around and speak to the responsibility for.

This empowers engineers to make markets, store vaults of obligations or guarantees, move assets as per guidelines given long before (like a will or a fates contract) and numerous different things that have not been developed yet, all without a go between or counterparty hazard.

2.4.2 Bitcoin

Bitcoin is a digital currency, a type of electronic money. It is a decentralized computerized money without a national bank or single chairman that can be sent from client to client on the shared bitcoin organize without the requirement for mediators.

Exchanges are confirmed by system hubs through cryptography and recorded in an open dispersed record called a blockchain. Bitcoin was designed by an obscure individual or gathering of individuals utilizing the name, Satoshi Nakamoto, and discharged as open-source programming in 2009. Bitcoins are made as a reward for a procedure known as mining. They can be traded for different monetary forms, items, and administrations. Research created by University of Cambridge evaluates that in 2017, there were 2.9 to 5.8 million special clients utilizing a digital currency wallet, the greater part of them utilizing bitcoin.

2.5 Smart Contracts

A smart contract is a PC code running over a blockchain containing a lot of standards under which the gatherings to that smart contract consent to interface with one another. In the event that and when the pre-characterized rules are met, the understanding is consequently implemented. The smart contract code encourages, confirms, and implements the exchange or execution of an understanding or exchange. It is the least complex type of decentralized robotization.

It is a system including computerized resources and at least two gatherings, where a few or the majority of the gatherings store resources into the smart contract and the advantages consequently get redistributed among those gatherings as per an equation dependent on specific information, which isn't known at the season of contract inception.

- A smart contract must be as smart as the general population coding considering all accessible data at the season of coding
- While smart contracts can possibly end up legitimate contracts if certain conditions are met, they ought not be mistaken for lawful contracts acknowledged by courts as well as law implementation. Be that as it may, we will likely observe a combination of legitimate

contracts and smart contracts rise throughout the following couple of years as the innovation turns out to be progressively full grown and boundless and lawful principles are received.

2.6 InterPlanetary File System (IPFS)

2.6.1 Overview

IPFS endeavors to address the inadequacies of the customer server model and HTTP web through a novel p2p document sharing framework. This framework is a combination of a few new and existing developments. IPFS is an open-source venture made by Protocol Labs, a R&D lab for system conventions and previous Y Combinator startup. Protocol Labs additionally creates corresponding frameworks like IPLD and File coin, which will be clarified underneath. Many designers around the globe added to the advancement of IPFS, so its arrangement has been an enormous endeavor. Here are the primary segments:

2.6.2 Distributed Hash Tables

A hash table is an information structure that stores data as key/value sets. In distributed hash tables (DHT) the information is spread over a system of PCs, and proficiently planned to empower productive access and query between hubs.

The fundamental points of interest of DHTs are in decentralization, adaptation to non-critical failure and versatility. Hubs don't require focal coordination, the framework can work dependably notwithstanding when hubs come up short or leave the system, and DHTs can scale to oblige a great many hubs. Together these highlights result in a framework that is commonly stronger than customer server structures.

2.6.3 Merkle DAG

A merkle DAG is a mix of a Merkle Tree and a Directed Acyclic Graph (DAG). Merkle trees guarantee that information squares traded on p2p systems are right, whole and unaltered. This

confirmation is finished by sorting out information squares utilizing cryptographic hash functions. This is just a function that takes an info and figures a one of a kind alphanumeric string (hash) relating with that input. It is anything but difficult to watch that an info will result in a given hash, yet extremely hard to figure the contribution from a hash.

2.7 Elastic Search

Elasticsearch can be utilized to look through a wide range of archives. It gives versatile inquiry, has close ongoing pursuit, and supports multitenancy. "Elasticsearch is dispersed, which implies that files can be isolated into shards and every shard can have at least zero reproductions. Every hub has at least one shards, and goes about as a facilitator to assign tasks to the right shard(s). Rebalancing and directing are done naturally". Related information is regularly put away in a similar record, which comprises of at least one essential shards, and at least zero reproduction shards. When a list has been made, the quantity of essential shards can't be changed.

Elasticsearch is created nearby an information accumulation and log-parsing tool called Logstash, an investigation and representation stage called Kibana, and Beats, a gathering of lightweight information shippers. The four items are intended for use as a coordinated arrangement, alluded to as the "Flexible Stack" (some time ago the "ELK stack").

Elasticsearch utilizes Lucene and endeavors to make every one of its highlights accessible through the JSON and Java API. It underpins faceting and permeating, which can be helpful for telling if new records coordinate for enrolled questions.

Elasticsearch (ES) is an appropriated and exceedingly accessible open-source internet searcher that is based over Apache Lucene. It's an open-source which is worked in Java in this manner accessible for some stages. You store unstructured information in JSON position which likewise makes it a NoSQL database. In this way, not at all like other NoSQL databases ES likewise gives web crawler functions and other related ElasticSearch Use Cases.

You can utilize ES for different purposes, two or three them given underneath:

- You are running a site that gives heaps of dynamic substance; be it a web based business site or a blog. By actualizing ES you cannot just give a vigorous web index to your web application however can likewise give local auto-complete highlights in your application.
- You can ingest various types of log information and after that can use to discover patterns and insights.

2.8 Logstash

2.8.1 Introduction

Logstash is an open source information gathering tool with ongoing pipelining abilities. Logstash can progressively bind together information from different sources and standardize your preferred information into goals. Wash down and democratize every one of your information for different progressed downstream investigation and perception use cases.

While Logstash initially drove advancement in log accumulation, its capacities expand well past that utilization case. Any sort of occasion can be enhanced and changed with a wide cluster of information, channel, and yield modules, with numerous local codecs further improving the ingestion procedure. Logstash quickens your bits of knowledge by bridling a more prominent volume and assortment of information.

2.8.2 The Power of Logstash

a. The ingestion workhorse for Elasticsearch and that's only the tip of the iceberg

On a level plane adaptable information handling pipeline with solid Elasticsearch and Kibana cooperative energy

b. Pluggable pipeline design

Blend, coordinate, and organize various information sources, channels, and yields to play in pipeline agreement

c. Network extensible and designer amicable module biological system

More than 200 modules accessible, in addition to the adaptability of making and contributing your own

2.9 Kibana

Kibana is an open source examination and representation stage intended to work with Elasticsearch. You use Kibana to hunt, see, and interface with information put away in Elasticsearch records. You can without much of a stretch perform propelled information examination and envision your information in an assortment of diagrams, tables, and maps.

Kibana makes it straightforward enormous volumes of information. Its basic, program based interface empowers you to rapidly make and share dynamic dashboards that show changes to Elasticsearch questions progressively.

Setting up Kibana is a snap. You can introduce Kibana and begin investigating your Elasticsearch files in minutes no code, no extra foundation required.

CHAPTER 3 METHODOLOGY

3.1 System Architecture

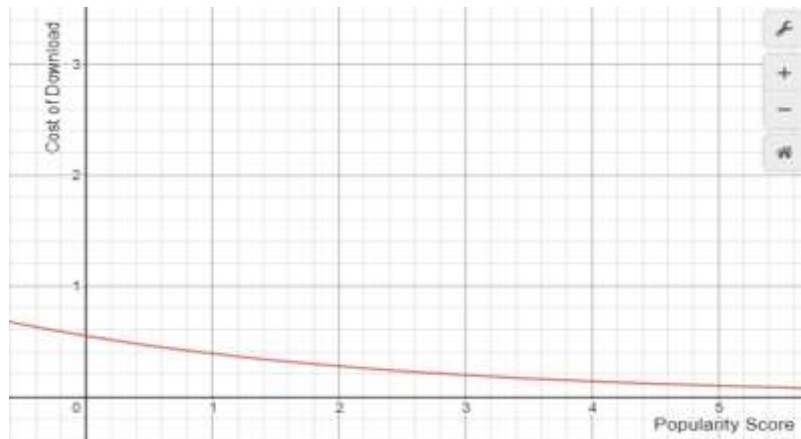
3.1.1 Currency Distribution Scheme

Through our platform, we want to incentivize the sharing of less available files. Thus payout for sharing less available but highly demanded files should be more than the payout for highly available files. We calculate the real-time payout and cost of download using the exponential decay function discussed below.

1. We need to determine the popularity score of every file in the system and keep updating it dynamically after any user interaction with the file. The **popularity score** for a file is defined as:

$$\frac{\text{count of leechers of the file} + \text{count of seeders of the file}}{2}$$

2. We need to calculate the cost of download and payout for upload for each file such that both of them decrease when popularity of the file increase. The cost of download is a decreasing exponential function of the popularity score. The most suitable function is $N_0 e^{-\lambda x}$ called the exponential decay function. Several natural decay processes such as radioactivity, rate of decrease of atoms in chemical reactions follow the same equation. It yields value N_0 when $x = 0$ and decreases exponentially henceforth [See image].



For the prototype, we have set the value of N_0 as $5.491 * 10^{-7}$ ethers and λ as $3.465 * 10^{-4}$ [Image not according to the values]. It would yield the seeder a gain of Rupees 11 per GB of data uploaded. In contrast, the average cost of data in India is Rupees 5 per GB.

A certain amount will also be kept in the contract which will be used to pay the developers and the verifiers (discussed later) for the maintenance of the network. Thus, the downloaders will be charged a bit more than the amount paid to the seeders. In return, the network will ensure that the downloaded files are malware free and completely safe to execute. We define the ratio R as:

$$R = \frac{\text{Payout for uploading the file}}{\text{Cost of download of the file}}$$

We have experimented for $R = 0.95$ and manage to maintain a positive balance in the contract.

3.1.2 Maintaining Security of Files

We will be dealing with the spread of malicious files by verifying the files being attempted to be shared by first running them and inspecting their behavior in a virtual environment. This is implemented through a quorum-based voting algorithm where the quorum members are also incentivized for their services. Set of most deserving 21 candidates will be elected to form the quorum. This method is inspired from the DPOS consensus algorithm (Larimer, 2017).

The quorum members can optimize the validation procedure such that it returns the most accurate results about the file's health. As discussed in the market opportunity section, we are well aware of the major types of malwares we may encounter. The verifiers will be pre-equipped of all the testing facilities. The information about the testing facilities ought to be public.

Based on the public information about the verifiers, users will vote much like the actual DPOS algorithm. The election process makes sure that the stakeholders (users) are ultimately in control because stakeholders lose the most when the network does not operate smoothly.

After all the verifiers have voted we check the fraction of verifiers who approve the file. If it passes a certain threshold, a content based hash would be created as File Link which would be used to uniquely identify the file. The File Size would be calculated and stored along with the rating. The file would be indexed in the database and will be available for downloading.

3.1.3 Maintaining Integrity of Files

Any change in the file would be detected before uploading as we are generating content based hashes of the file. Thus, if a seeder with malicious intent tries to modify any or whole of the file, it will change the hash of the resulting file. We will match the hash of the file being uploaded with the hash of the file in the system. If the file is modified in any way, it will not be identified by the client software and uploading would not begin in that case.

3.1.4 Searching for Content on Client Side

Searching for the required file in a large public database is a daunting task. To facilitate searching on the client side application, we need to have a database in indexed form. We have used Elasticsearch and logstash to simplify the task. The results of the search would be in increasing order of Levenshtein distance from the query (character mismatches). We are currently working on ways to improve search (see improving the product section).

3.1.5 Managing Uploading on Client Side

For the prototype version, we have used Transmission's command line interface and Transmission remote package to compute the data exchange happening on the client's machine. Transmission is essentially a free software program to manage exchange of data packets from multiple peers at once on Linux distributions.

The client software runs a script every 10 seconds which determines the difference of the total uploaded data between current timestamp and the previous timestamp for every file. This means that the user will be paid every 10 seconds using payments enabled by the Ethereum blockchain.

3.1.6 Creation of New File

When a user clicks the Create New File button, he can add a new document of any type from his computer to our network. The file is directly transferred to the 21 verifiers who calculate the file size. These verifiers then test the files in a virtual environment on their machine and submit their verdict on the health of the file. The final health score assigned to the file would be mean of all the individual verifier scores.

3.2 Smart Contract Architecture

3.2.1 Files Base

The contracts describe the available Files on the platform. The variables in the structure are:

- | | |
|--------------------------|--|
| 1. FileLink | unique content based hash of the file. Acts as primary identifier |
| 2. FileName | name of the file, as visible to the users. Indexed by Elasticsearch. |
| 3. Rating | contains overall health score of the file. |
| 4. FileSize | an integer storing size of the file in Kbs |
| 5. CostToDownload | an integer storing the cost in milliEther |
| 6. Creator | stores address of the creator of file. |

It contains two simple functions:

- | | |
|--------------------------|--|
| 1. CreateFileLink | called when a user adds a new file. Rating initialised to 0. |
| 2. GetFileInfo | returns all the file information when fileLink is given. |

3.2.2 Users Base

This contract describes the users on the network. The variables in the user structure are:

1. **Mapping from user address to user**
2. **Mapping from user to its downloads**
3. **Mapping from user to its uploads**

It contains the following functions:

1. **UserExists** checks if sender address is present in our database
2. **CreateUser** creates a new user object if sender address is new
3. **StartDownload** payable method to accept payment and initiate download
4. **AddToDownloads** signals the client software when download starts
5. **StartUpload** start uploading the file
6. **RecieveRewards** adds money to user wallet when called
7. **GetDownloadSize** utility function to get total downloaded size
8. **GetUserDownloadInfo** read function to return all downloaded user files

It contains two arrays having indexes of the user's downloaded and uploaded files. A userExists function checks sender's address in the database whenever a user enters our system or initiates a download. If the address is new, it calls createUser() function. Thus user identity is always checked and one cannot login from someone else's account.

3.3 Development Environment and Tools

3.3.1 Development Tools

1. **Linux** - Linux is a family of open source Unix-like operating systems based on the Linux kernel
2. **Visual Studio Code** - It is a source-code editor developed by Microsoft for Windows, Linux and macOS. It was used throughout during the development stages.
3. **NodeJS and NPM**- NodeJS is server side Java Script platform to create application that interacts with our ethereum node. NPM stands for Node Packet Manager which provides tools and libraries to generate for ethereum.
4. **Rinkeby Testnet** - A proof-of-authority blockchain, started by the Geth team. It simulates the actual Ethereum blockchain to provide developers a testing platform.

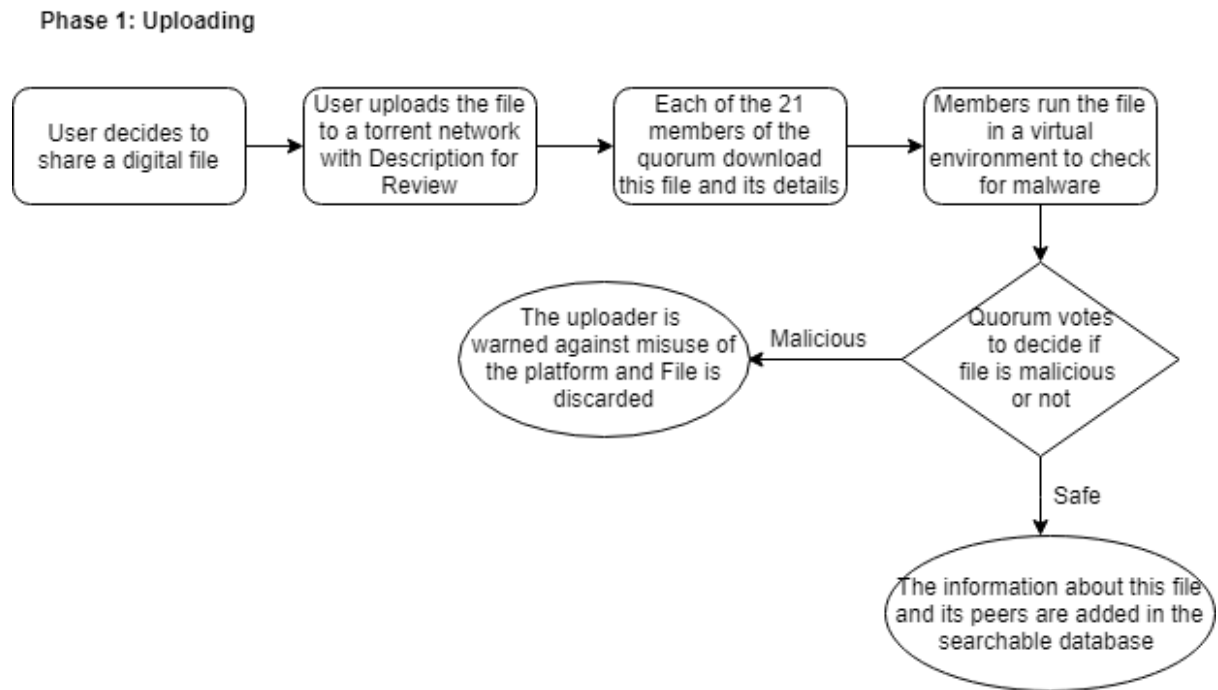
5. **Metamask-** Metamask is a plugin for browsers which allows to make ethereum transactions. It helps in bridging the gap between user interface for ethereum (Dapps) and the normal regular web (Google Chrome).
6. **Truffle** - Truffle is used to compile, test, build smart contracts and provides a development framework to increase speed in the development process.
7. **Transmission CLI** – Used to control Transmission bit-torrent client using console commands
8. **Elastic Search** - Elasticsearch is a search engine based on the Lucene library. It provides a distributed, multitenant-capable full-text search engine with an HTTP web interface and schema-free JSON documents.
9. **Logstash** - *Logstash* is an open source, server-side data processing pipeline that ingests data from a multitude of sources simultaneously, transforms it, and then sends it to your elastic search.

3.3.2 Languages/Frameworks Used

1. **Solidity** - Solidity was used to implement and deploy the smart contract on the Ethereum blockchain.
2. **Python** - Python was used to run network simulations to gain insights about the expected results.
3. **JavaScript** – JavaScript was used to develop the client software.
4. **ReactJS** – ReactJS was used to create interactive UIs.
5. **HTML** – HTML was used for designing web pages on client side.
6. **CSS and semantic-ui:** They were used to style the webpages on client side.

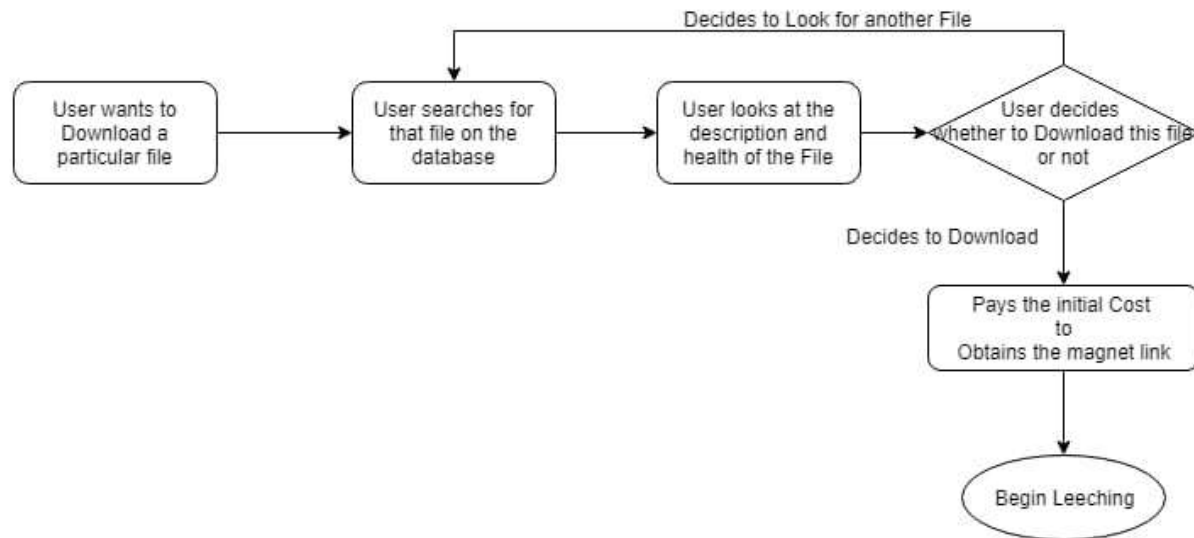
3.4 Data Flow Diagram

Phase I: Adding a new file/Uploading



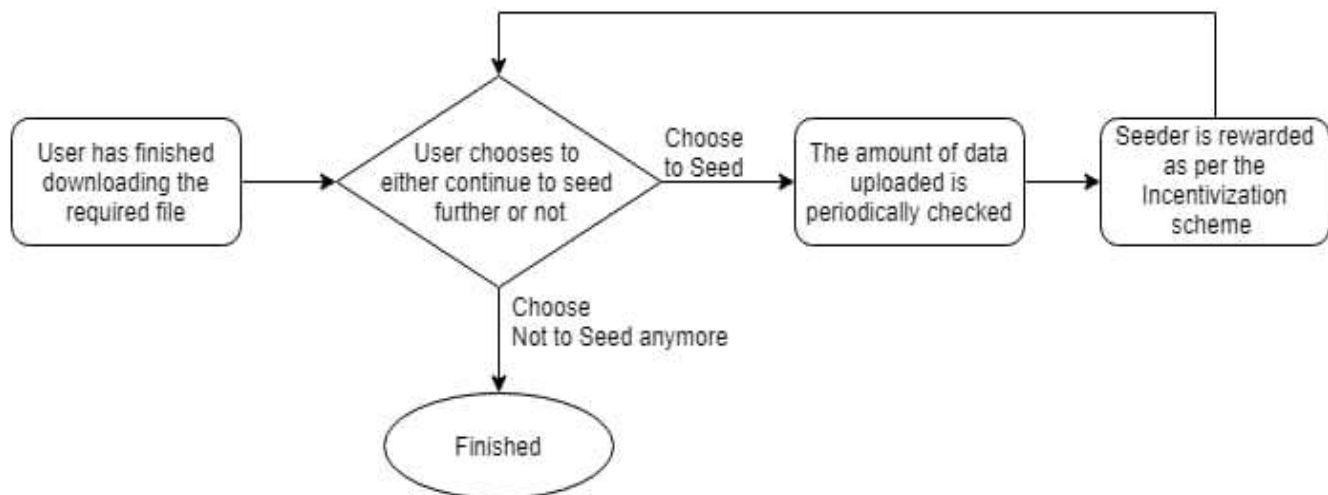
Phase II: Downloading/Leeching

Phase II: Downloading



Phase III: Seeding

Phase III: Incentivised Seeding



3.5 Contract Code Snippets

```
1  pragma solidity ^0.4.17;
2
3  contract BTorrent{
4
5      struct Torrent{
6
7          string fileLink;
8          string fileName;
9          uint16 userRating;
10         uint16 fileSize;           // In Kbs as of now
11         uint16 costToDownload;    // in milliEther
12         address creator;
13     }
14
15     mapping(string => Torrent) torrentLinkToInfo;
16
17     function createTorrent(string fileLink, string fileName, uint16 fileSize) public {
18
19         Torrent memory newTorrent = Torrent({
20             fileLink: fileLink,
21             fileName: fileName,
22             userRating: 0,
23             fileSize: fileSize,
24             costToDownload: fileSize/1000,
25             creator: msg.sender
26         });
27
28         torrentLinkToInfo[fileLink] = newTorrent;
29         return;
30     }
31
32
33     function getTorrentInfo(string fileLink) public view returns (string,string,uint16,uint16,uint16,address){
34
35         Torrent memory newTorrent = torrentLinkToInfo[fileLink];
36         return (newTorrent.fileLink, newTorrent.fileName, newTorrent.fileSize,
37             newTorrent.costToDownload, newTorrent.userRating, newTorrent.creator);
38     }
39 }
40
41
42 contract UserBase{
43
44     struct Torrent{
45
46         string fileLink;
47         string fileName;
48         uint16 userRating;
49         uint16 fileSize;           // In Kbs as of now
50         uint16 costToDownload;    // in milliEther
51         address creator;
52     }
```

```

54 * struct User{
55     mapping(uint16 => Torrent) downloads;
56     uint16 downloadSize;
57 }
58
59 mapping(address => bool) isUser;
60 mapping(address => User) addressToUser;
61
62
63 * function userExists(address UserID)public view returns (bool){
64     return isUser[UserID];
65 }
66
67 * function createUser() public{
68     isUser[msg.sender]= true;
69 *     addressToUser[msg.sender] = User({
70         downloadSize:0
71     });
72     return;
73 }
74
75
76 * function startDownload(uint16 costToDownload, string fileLink, string fileName, uint16 fileSize,address creator, address UserId) public payable {
77
78     require(msg.value >= costToDownload);
79     addToDownloads(fileLink, fileName, fileSize, creator,UserId);
80     return;
81 }
82
83 * function addToDownloads(string fileLink, string fileName, uint16 fileSize,address creator, address UserId) public{
84
85 *     Torrent memory torrent = Torrent({
86         fileLink: fileLink,
87         fileName: fileName,
88         userRating: 0,
89         fileSize: fileSize,
90         costToDownload: fileSize/1000,
91         creator: creator
92     });
93
94     User storage currUser = addressToUser[UserId];
95     currUser.downloads[currUser.downloadSize]=torrent;
96     currUser.downloadSize = currUser.downloadSize+1;
97
98     return;
99 }
100
101
102 * function receiveReward(uint16 uploadedData, address UserId) public {
103     UserId.transfer(uploadedData /1000);
104     return;
105 }

```

CHAPTER 4 REPORT ON PRESENT INVESTIGATION

We successfully implemented the incentivization scheme through a smart contract and demonstrated the transfer of a file on our system

Search for file name

the|

★ The Maltese Falcon

★ The ABCs of Death

★ The Abandoned

★ The Abdication

★ The Abduction Club

★ The Abductors

★ The Abominable Snowman

★ The Abyss

★ The Accidental Husband

★ The Accidental Tourist

★ The Accountant

★ The Accused

★ The Ace of Hearts

★ The Acid Eaters

★ The Acid House

User Address: 0xB762F06fc8951EA29673835f77E70b6866db4Fd9

Downloads

Uploads

Torrent files can be searched by typing the name of the file, this dataset is indexed using Elastic Search and will be hosted on a Centralized server for faster query processing

Search for file name

- ★ [Hunter Gatherer](#)
- ★ [Hunter Prey](#)
- ★ [Hunger Point](#)
- ★ [Haunter](#)
- ★ [Hunt for the Labyrinth Killer](#)
- ★ [Hunt for the Wilderpeople](#)
- ★ [Hunt to Kill](#)
- ★ [Hundra](#)
- ★ [The Heart is a Lonely Hunter](#)
- ★ [The Lion Hunters](#)
- ★ [Cry of the Hunted](#)
- ★ [The Eagle Huntress](#)
- ★ [Aqua Teen Hunger Force Colon Movie Film for Theaters](#)
- ★ [The Hunt for Red October](#)
- ★ [The Hunting of the President](#)

User Address: 0xB762F06fc8951EA29673835f77E70b6866db4Fd9

Downloads

Torrent File Information

- **FileName:** [The Maltese Falcon](#)
- **Link of File:** [magnet:?xt=urn:askj](#)
- **Size of File:** [950 KB](#)
- **User Ratings:** [3 \(0-5\)](#)
- **Cost of Downloading:** [1583 Wei](#)

Uploads

Torrent File Information

- **FileName:** [The Maltese Falcon](#)
- **Link of File:** [magnet:?xt=urn:askj](#)
- **Size of File:** [950 KB](#)
- **User Ratings:** [3 \(0-5\)](#)
- **Cost of Downloading:** [1583 Wei](#)

Torrent File Information

- **FileName:** [Batman and Robin](#)
- **Link of File:** [magnet:?xt=urn:elvd](#)
- **Size of File:** [1142 KB](#)
- **User Ratings:** [4 \(0-5\)](#)
- **Cost of Downloading:** [1903 Wei](#)

Torrent File Information

- **FileName:** [Batman and Robin](#)
- **Link of File:** [magnet:?xt=urn:elvd](#)
- **Size of File:** [1142 KB](#)
- **User Ratings:** [4 \(0-5\)](#)
- **Cost of Downloading:** [1903 Wei](#)

The right hand side contains the user's Ethereum account address. It also contains two lists, first list contains past downloaded files and the other list indicates the files which are available for seeding. The list contains Filename, File Link, Size of File, User Ratings and the Cost of downloading the file

Search for file name

india

Torrent Download Information

- FileName: *Indiana Jones and the Kingdom of the Crystal Skull*
- Link of File: <magnet:?xt=urn:btih>
- Size of File: 1000 Kb
- User Ratings: 4 (0-5)
- Cost of Downloading: 1666 wei

Start Download

User Address: 0xB762F06fc8951EA29673835f77E70b6866db4Fd9

Downloads

Torrent File Information

- FileName: *The Maltese Falcon*
- Link of File: <magnet:?xt=urn:askj>
- Size of File: 950 KB
- User Ratings: 3 (0-5)
- Cost of Downloading: 1583 Wei

Uploads

Torrent File Information

- FileName: *The Maltese Falcon*
- Link of File: <magnet:?xt=urn:askj>
- Size of File: 950 KB
- User Ratings: 3 (0-5)
- Cost of Downloading: 1583 Wei

Torrent File Information

- FileName: *Batman and Robin*
- Link of File: <magnet:?xt=urn:elvd>
- Size of File: 1142 KB
- User Ratings: 4 (0-5)
- Cost of Downloading: 1903 Wei

Torrent File Information

- FileName: *Batman and Robin*
- Link of File: <magnet:?xt=urn:elvd>
- Size of File: 1142 KB
- User Ratings: 4 (0-5)
- Cost of Downloading: 1903 Wei

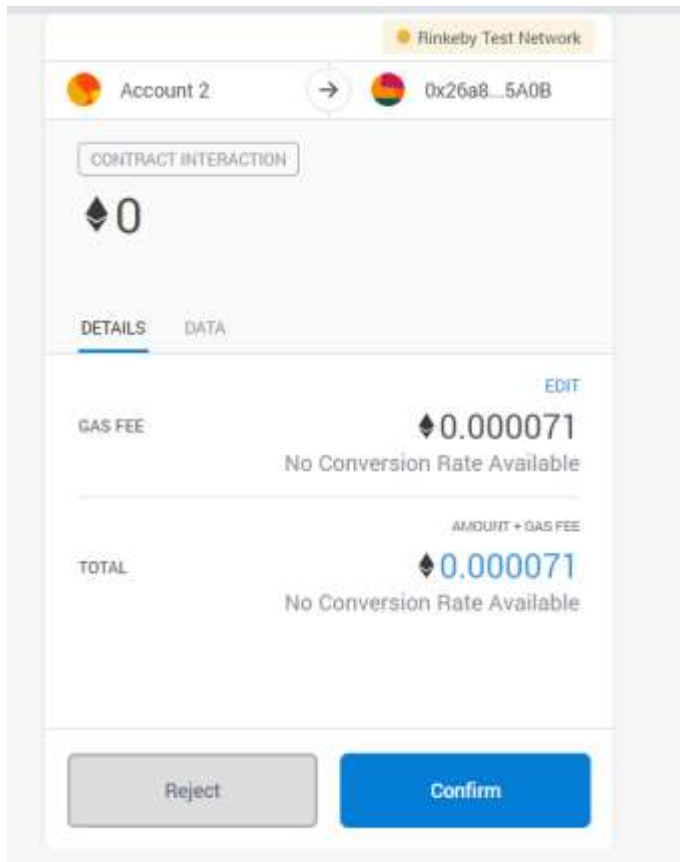
Torrent File Information

- FileName: *Hunter Gatherer*
- Link of File: <magnet:?xt=urn:sjpl>
- Size of File: 2024 KB
- User Ratings: 4 (0-5)
- Cost of Downloading: 3373 Wei

Torrent File Information

- FileName: *Hunter Gatherer*
- Link of File: <magnet:?xt=urn:sjpl>
- Size of File: 2024 KB
- User Ratings: 4 (0-5)
- Cost of Downloading: 3373 Wei

The download process can be started by selecting the desired file from the search window and then clicking on Start Download button



After Clicking on Start Download Button a MetaMask notification will pop up indicating the total amount of ether which will be charged for this transaction

Search for file name

the

User Address: 0xB762F06fc8951EA29673835f77E70b6866db4Fd9

Downloads

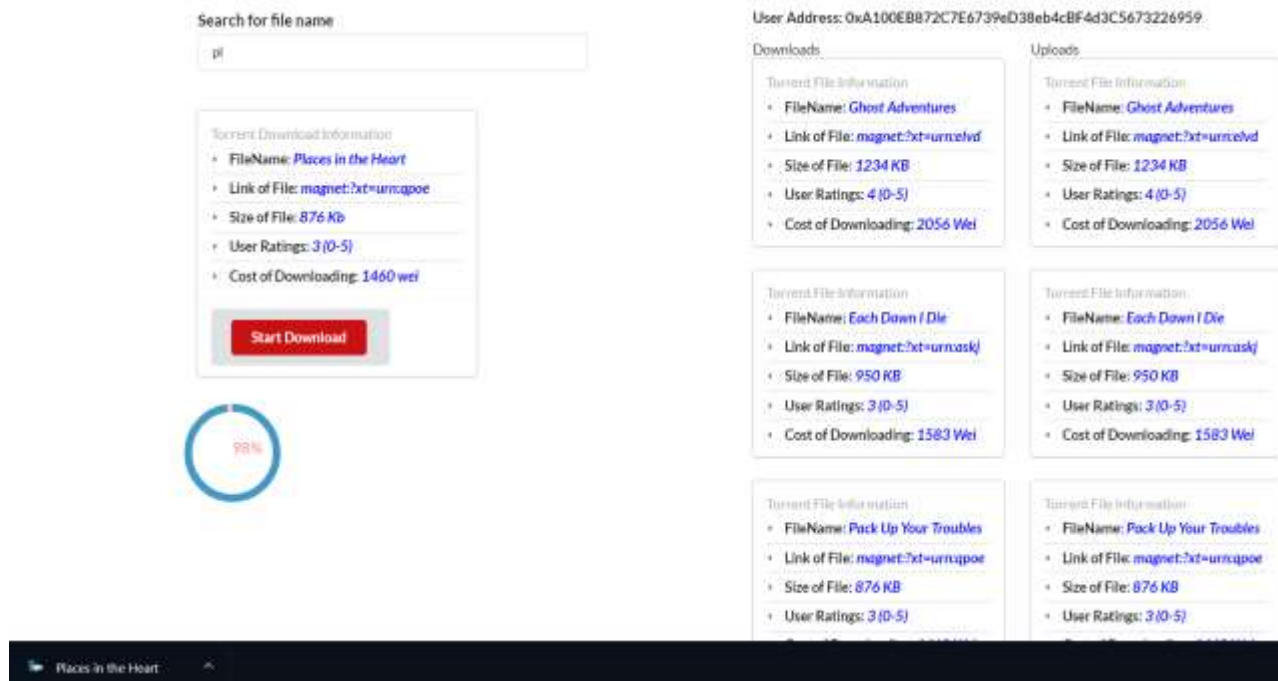
Uploads

Torrent Download Information

- **FileName:** [The Maltese Falcon](#)
- **Link of File:** [magnet:?xt=urn:askj](#)
- **Size of File:** 950 Kb
- **User Ratings:** 3 (0-5)
- **Cost of Downloading:** 1583 wei

Start Download





After confirming the transaction the download process will initiate using Transmission Remote. A progress bar will keep track of the download status of the file

Contract Overview

Balance0.0000000000000000472 Ether

More Info

My Name TagNot Available

Contract Creator0xa100eb672c7e67... at bn 0xb796031174e67a8...

Transactions

Code

Events

IF Latest 7 txns







Txn Hash	Block	Age	From	To	Value	[Txn Fee]
0x0e061977250b02...	4384882	44 secs ago	0x4148f8b0c38410b...	0x26a88d92a96890...	0.0000000000000003 Ether	0.001127116
0x09c529c4cc074...	4384876	2 mins ago	0x4148f8b0c38410b...	0x26a88d92a96890...	0 Ether	0.00394723
0x09a00a207d25ea...	4384869	3 mins ago	0xa100eb672c7e67...	0x26a88d92a96890...	0.0000000000000001 Ether	0.001127382
0x343705aaed598a...	4384857	6 mins ago	0xa100eb672c7e67...	0x26a88d92a96890...	0.0000000000000001 Ether	0.001127382
0x5ea12173079c9d3...	4384846	9 mins ago	0xa100eb672c7e67...	0x26a88d92a96890...	0.0000000000000002 Ether	0.001127382
0x09cb313d7778083...	4384834	12 mins ago	0xa100eb672c7e67...	0x26a88d92a96890...	0 Ether	0.00394723
0xb796031174e67a8...	4384828	14 mins ago	0xa100eb672c7e67...	888 Contract Creation	0 Ether	0.001732116

Since this transaction occurs on the Rinkeby network users can view actual log and details of their transaction on official Rinkeby website, hence all the exchange will be completely transparent and visible to the user

Transaction Details

Overview

[This is a Rinkeby **Testnet** Transaction Only]

Transaction Hash:	0x6fdeb0e297d25eacbd9f8efb843d3d890cf1621532948ab20c30e2859a8a8905 
Status:	 Success
Block:	4384869  16 Block Confirmations
Timestamp:	⌚ 4 mins ago (May-15-2019 06:41:50 AM +UTC)
From:	0xa100eb872c7e6739ed38eb4cbf4d3c5673226959 
To:	Contract 0x26a88d92a968907b339be4cd972f1b13a1ab5a0b  
Value:	0.000000000000000146 Ether (\$0.00)
Transaction Fee:	0.000112702 Ether (\$0.000000)

[Click to see More](#) 

This indicates the details about the transaction including its Status, Timestamp, Transaction Fee, Sender and Receiver Account Address etc.

CHAPTER 5 CASE STUDY

5.1 Running a Network Simulation

5.1.1 Popularity Scores

To gain better insights about the expected currency distribution, we ran a simulation of the network in Python. Actual data containing 28000 files was scraped from a popular P2P site. The files were assigned 3 types of popularity score:

1. **Popularity score 1:** It gave more weightage to seeders count.

Formula: $0.7 * \text{seeders} + 0.3 * \text{leechers}$

2. **Popularity score 2:** It gave more weightage to leechers count

Formula: $0.3 * \text{seeders} + 0.7 * \text{leechers}$

3. **Popularity score for currency distribution:** It gave equal weightage to seeders and leechers count.

Formula: $0.5 * \text{seeders} + 0.5 * \text{leechers}$

2800 (10% of the number of files) users were simulated. For every user, a number between 50 to 100 was selected and same of number of files were selected randomly and distributed to them. The distribution will be according to popularity score 1 i.e. files with more popularity score will have higher chances of selection. These are the files they will be uploading later. The same procedure will be followed to distribute the files that are required by the users but this time popularity score 2 will be used.

The distribution of currency will be governed by the function $N_0 e^{-\lambda x}$. The value of x here would be Popularity score for currency distribution which is mean of seeders and leechers count.

5.1.2 Parameters

1. **N_0 :** The cost when $x = 0$ or the rarest possible a file can become.
2. **K :** The popularity score at which the cost would fall to $\frac{1}{2}$ of N_0 .
3. **λ :** Coefficient of x in the exponent.
4. **P :** A file with popularity score 100 will cost P paisa per MB to download.
5. **R :** Ratio of cost of download to the payout.

For every file, we note the users who require it and the users who already have it. We select a random set of users to seed the files. The seeders and leechers are matched. The cost of download and the payout are calculated by the above formula and will be deducted and distributed accordingly.

5.1.3 Data Summary

1. The torrent data only consisted of the latest files added to the network no later than 5 days from 20th April 2019.
2. Mean size of the files: 1663.98 MB
3. Standard Deviation of size of the files: 6714.68 MB

5.1.4 Test Values:

1. **K :** 2000
2. **P :** 1 paisa per MB
3. **R :** 0.95
4. **N_0 :** $5.491 * 10^{-7}$ ethers (derived from K and P)
5. **λ :** $3.465 * 10^{-4}$ (derived from K and P)

5.1.5 Observations:

1. The value of money in contract increases as we decrease K . It means that
2. Users with maximum profit have gained more in absolute value than users with maximum loss. It means that people, on an individual level, gain more money than losing.
3. About 2.2% of the files requests could not be satisfied because of randomness in initial distribution.
4. Variance in currency distribution was 0.04131 ETH for $K = 100$, $P = 1$ and $R = 0.95$.
5. Around 55% of people (or 1500) ended up with negative amount of ether in their wallet. It means users must optimize their uploading to gain from the network. This prevents our platform from becoming a completely passive money making platform, which is a good thing.

CHAPTER 6 RESULTS

We have successfully achieved the following objectives through our work as demonstrated in the report on present investigation and case study. Given below is a comparison between our platform and Steemit platform

6.1 SteemIt Versus Our Platform

At first glance, it seems as if our model was very similar to the popular blogging platform SteemIt. We found out that there are many dissimilarities between the two and are listing some of them here:

1. **In SteemIt, the users have an option to never interact with the content.** SteemIt users can be readers only. In our system, a participant has to pay a fee to download the content. There can be no passive participants in our system. This leads to the demand for high quality, malware free content.
2. **A content provider in SteemIt ought to be the content creator also.** He/She gets paid proportional to the community's perception of the quality of the content. In our system, the content provider does not need to create original content. The payout is pre-decided and determined by the algorithm. This makes his job less dependent upon chance. Thus, he may pre-plan and optimise uploading according to his wish.
3. **SteemIt maintains a price feed** which helps stabilise the dollar value of the Steem currency. Our prototype operates upon Ethereum directly thus we need not maintain any price feed.
4. **In Steemit, payments are dependent upon quality of content.** Value in SteemIt resides in the blogs. The payouts are dependent upon the quality of the content. Whereas in our system, the value resides in the files being exchanged. The various costs are proportional to the size and popularity of the files irrespective of the type or quality.

6.2 Future Work and Improvements

We have thought upon many features and optimisations which can help improve user experience on our platform. Some of them can make the platform more robust as well as reduce the latency time.

1. **Developing a real-time recommendation engine for the uploaders.** A regular uploader may have several constraints like upload speed, cost of upload etc. We plan to design a recommendation engine which can help optimise the monetary gain per unit time or per unit data consumed whatever is the seeder's choice. The various parameters which affect the recommendation of a user are:

- a. Cost of uploading
- b. FUP limit on high speed data (if any)
- c. Payout per file of already downloaded files
- d. Number of other nodes currently uploading already downloaded files

Working: The real-time data about the downloaded files can be fetched from the blockchain. This would not take much resource as it is a read-only query. The computation can be done on client software itself after feeding cost and limit of uploading into the algorithm.

2. **Making search more efficient.** There are two possible architectures to improve search:

- a. Store the content addressable hashes and sizes of the files on central servers much like Bittorrent websites Kickass and PirateBay. The real-time payout and cost of downloads can be stored on the blockchain so that they can be trustworthy.
- b. Store all the information on blockchain and try to improve the search architecture. This would make file storage truly decentralised with the overhead of space on blockchain which is costly in the present time. This would also minimize user inconvenience and would drastically improve our UX.

3. **Devising a better function for computing various costs.** Costs analysis is perhaps the most important aspect of our project. We are currently experimenting with several simulations of the network to devise metrics to evaluate various network characteristics. Then we shall apply optimization techniques like regression and curve fitting to attain most optimal value of N_0 and λ .

References

1. Megaupload: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=2176246
2. Optimal selfish mining strategies in bitcoin:
https://fc16.ifca.ai/preproceedings/30_Sapirshtein.pdf
3. Performance analysis of bittorrent protocol:
https://www.researchgate.net/publication/261269751_Performance_analysis_of_BitTorrent_protocol
4. Big torrent measurement:
https://www.researchgate.net/publication/326276604_Big_torrent_measurement_A_country-network-and-content-centric_analysis_of_video_sharing_in_BitTorrent/amp
5. Joystream news: <https://news.bitcoin.com/joystream-test-drive-how-to-get-paid-seeding-bittorrent-files/>
6. Understanding and Improving Ratio Incentives in Private Communities:
<https://dl.acm.org/citation.cfm?id=1846245>
7. Incentives build robustness in bittorrent: <http://bittorrent.org/bittorrentecon.pdf>
8. P2P networking with bittorrent: <http://web.cs.ucla.edu/classes/cs217/05BitTorrent.pdf>
9. DPOS: <https://steemit.com/dpos/@dantheman/dpos-consensus-algorithm-this-missing-white-paper>