DeStore: A Novel Approach to Decentralized Cloud Storage through Peer to Peer Networking using Blockchain

ABSTRACT

DeStore, standing for Decentralized Cloud Storage, is a novel approach to creating a decentralized solution for efficient, fast, and secure access to cloud storage by implementing the concept of Blockchain universal ledger in an open Peer to Peer Network. It aims to bring more reliable, cheaper and faster cloud access to people at remote locations from the data centers, who cannot utilize server-based cloud solutions even when connected to the internet grid, due to very high latency, poor internet downlink speeds and packet loss due to signal degradation. It is an open Peer to Peer Networking Solution that utilizes the unused space of the peers' hard drives to securely store data fragments, which have been encrypted using the AES-256 standard Encryption protocol. Data integrity and security is ensured by the Blockchain model, where retrieving data fragments is verified by the peers and a block is added to the universal data access ledger for every successful data retrieval.

NOVELTY OF THE PROJECT

The novelty lies in the idea to store data in fragments in a truly distributed network instead of the existing traditional centralized cloud servers and using blockchain ledger to validate data integrity. Blockchain, traditionally only applied for cryptologic purposes, has been put to innovative use to preserve authority in a P2P network data storage, which otherwise often fails due to lack of enough security and integrity measures.

- Exploring the features of a P2P network to do more than just share data, we use the P2P network to effectively decentralize data storage solutions which resolves current security breaches that come with the centralization of data. Going much more beyond than encryption, the concept of decentralization makes the data storage network faster than client/server unidirectional requests, fault-tolerant to traditional client/server cloud computing or network topology fails, and immune to Distributed Denial-of-Service (DDoS cyber attacks) that cloud data storage servers are critically vulnerable to.
- The fault of P2P network that compromises security and integrity of data, is overcome by the novel application of blockchain, where every data transaction, after being verified by all the peers, is added to the ledger as a universal record of all data transaction, ensuring transparency, security against malicious peers, and integrity of data.
- Taking an innovative approach to the existing transfer protocols, we have designed a multi-threaded fragment transfer protocol using Socket programming for faster and more efficient file fragment transfer over the network.

BACKGROUND RESEARCH

- In the current networking scenario the client/server model for cloud-based storage, besides being expensive, is inefficient for long distance communication, suffers downtime, tends to bottleneck[1] when multiple users try to extract uploaded files at the same time and suffer significantly from the "Last Mile Problem".

- As the number of users ("noisy neighbors")[2] increases, the performance of the server decreases as those connections suffer from very high latency, packet loss due to signal degradation, and are bottle necked by their connection thereby resulting in a very poor downlink speed to the cloud. The major sufferers are people living in the village who despite having procured an expensive internet connection, from the government, cannot utilize any cloud storage based application.

- Problems with the centralization of data: Centralization of cloud storage makes data less accessible to people living far away, decreases reliability as servers tend to crash and run into major troubles as time progresses. Cases of full data erases and isolation tends to happen due to successive drive crashes, geographic disasters, and major hardware failures. Even extremely reliable companies with data centers on every continent, and features such as geo-replication, lose large data sets once a while as happened on Apr. 28, 2011 where Amazon's cloud crash[3] disaster permanently destroyed many customers' data. The data loss was apparently small relative to the total data stored, but anyone who runs a website can immediately understand how terrifying a prospect any data loss is. This is catastrophic in the case of financial units and banks.

- Companies increasingly store sensitive data in the cloud to use it in case disaster recovery situation. When a cloud server is breached, cybercriminals can gain access to this sensitive data. Centralization causes all the data to be stored on one system while Decentralization improves distributed storage. A glaring example of this security failing is the attack on Facebook's metadata[4] and user information leak, which could have been prevented in a decentralized network, with power equally distributed among all users.

- When a DDoS attack is made onto a server through a victim that is an unknowing client, a massive number of requests are made to the server. The cloud allocates full resources due to its auto scalability functions, while the users, who are allocated on the same server are unable to access the cloud when the DDoS attack is made due to flooding of the server with a large number of false traffic.

- Fault Intolerance: In the current scenario servers are quintessential in all parts of the network and Peer to Peer connections as a whole are completely ignored except for very small-scale LAN. If any of the servers go down, the company, as well as the clients, are severely affected.

ALGORITHM

The project algorithm is divided into the following modules for implementation-

1. Fragment Size Prediction with Pareto's efficiency-graph

Pareto efficiency is a position in where no improvement can be done to one factor without indirectly deteriorating the other interconnected factor.[5]

How the File fragments size and distribution should be, with respect to the number of Users online, is predicted with the help of Pareto's efficiency graph. To determine file size a Pareto's distribution graph is computationally determined by all the peers by the data written IP address table by DHT.

From these predicted graph points are determined where network efficiency and redundancy is maximum.

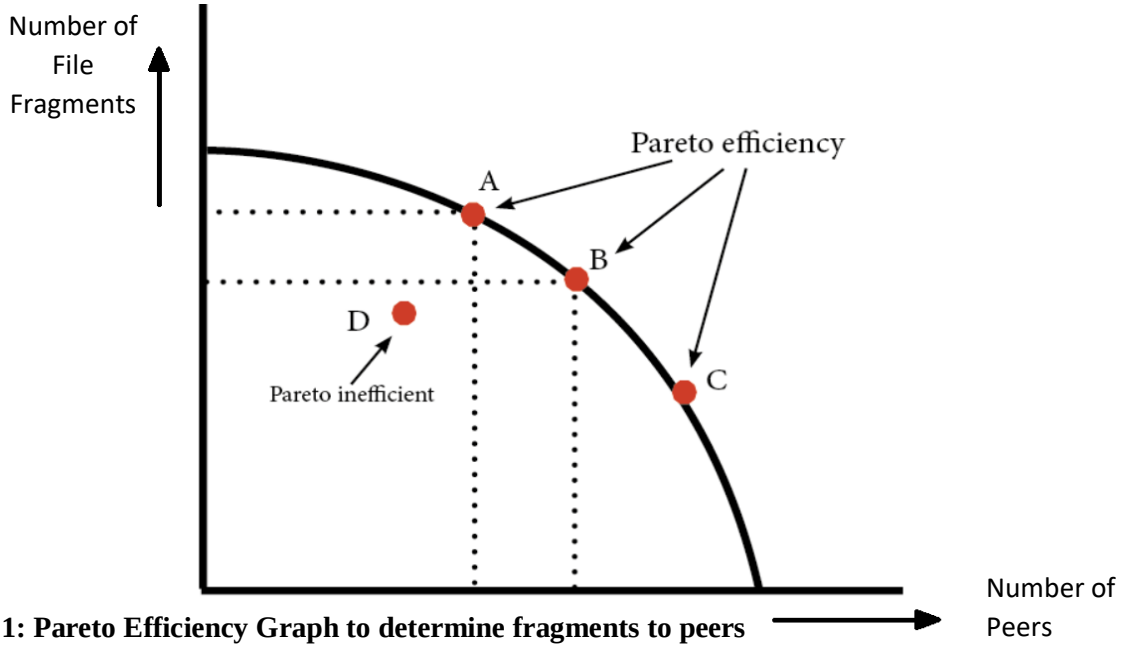


Fig 1: Pareto Efficiency Graph to determine fragments to peers

We determine the Pareto frontier or Pareto set for our given system of P2P Network, which is the set of parameterizations (allocations) that are all Pareto efficient. By finding all potential optimal solutions, we can design our network with the number of required peers for efficiency within these constraints.

The Pareto frontier $P(Y)$ is:

$$P(Y) = \{ y' \in Y : \{ y'' \in Y : y'' > y', y'' \neq y' \} = \Phi \}$$

Here Y is the set of IP addresses from the IP address list where n(Y) is the number of peers connected and y', y", y"', … are all elements of this IP address list.

We have incorporated this function in our designed program to predict optimal values of this set.

- There are infinite number of possibilities in which a file could be fragmented. These possibilities are the elements of a set known as Y.
- This set is passed through the function above and from the return value i.e. the Pareto's set, 50 random values are taken and represented on these graphs. Blue dots represent optimal methods while red dots represent suboptimal methods.
- With the help of this graph, the best fit curve is plotted and points on this best fit curve represent almost Pareto efficient methods to fragment the data.

Therefore we estimate the number of fragments into which the file has to be broken down into so that the network is faster and more efficient.

2. Redundancy in File-sharing using Erasure Codes

When peer nodes join and leave the peer network, they do so at a rate called the churn rate. Because of churn and temporal inaccessibility, a node cannot rely on peers to be available at a particular time in the future, therefore data must be stored in peers in a redundant way, i.e. the data placement at peers must be organized such that in case of loss of a certain fraction of the peers, the data can still be recovered.

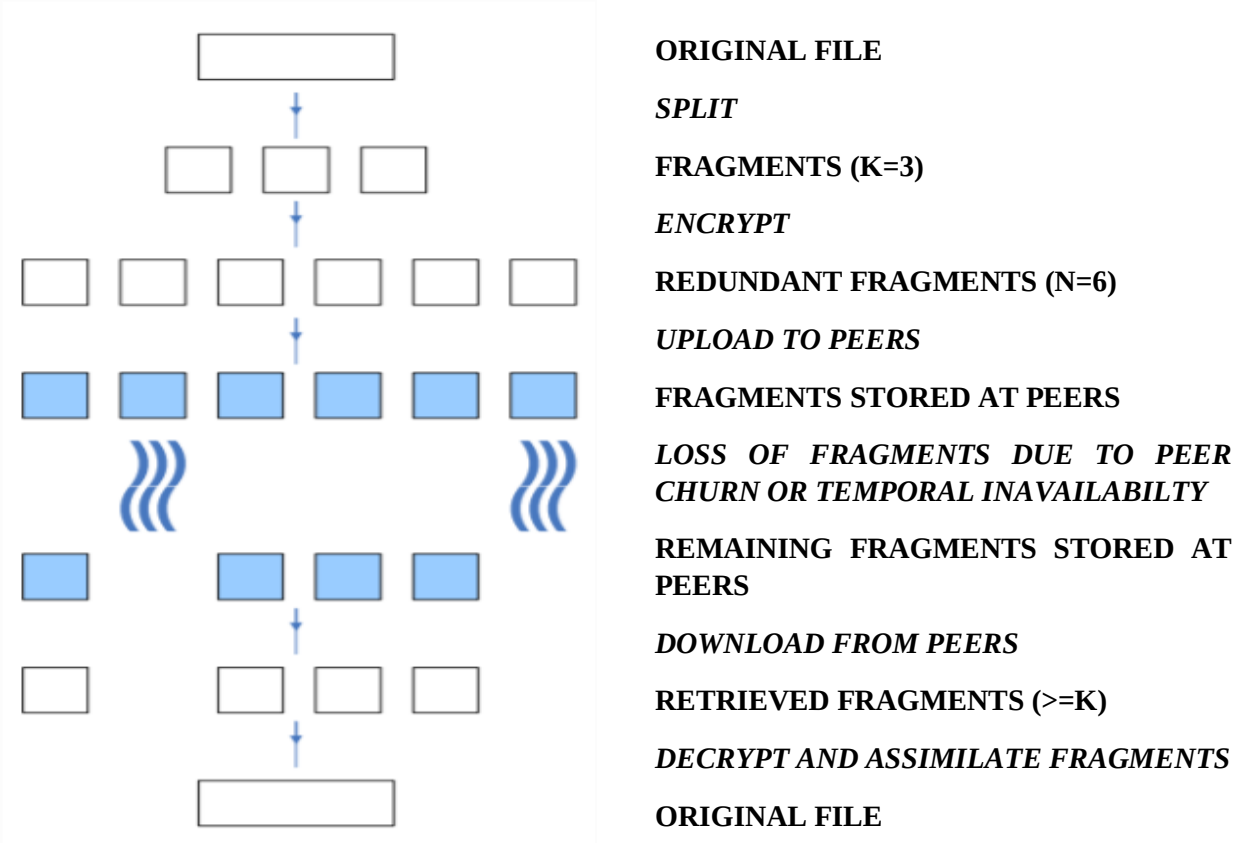


Fig 2: The complete cycle of File Storage and Retrieval in our P2P Network

We apply erasure codes to the file data as error correction codes, to compensate for errors occurring during the transmission of data. In this method, the original data is encoded into a longer message such that only a fraction of the data suffices to reconstruct the original data at the receiving end.

When applying erasure codes to storage, a file is first divided into k fragments and then encoded into n fragments, with k < n and a constant fragment size. The ratio n/k indicates how much extra storage is used compared to the original file size. The main property of erasure codes is that any k fragments are needed to restore the original file.

With similar storage requirements and peer availability, the usage of erasure codes delivers a much higher probability of successful data restore than straightforward replication[6].

3. Implementing the Blockchain

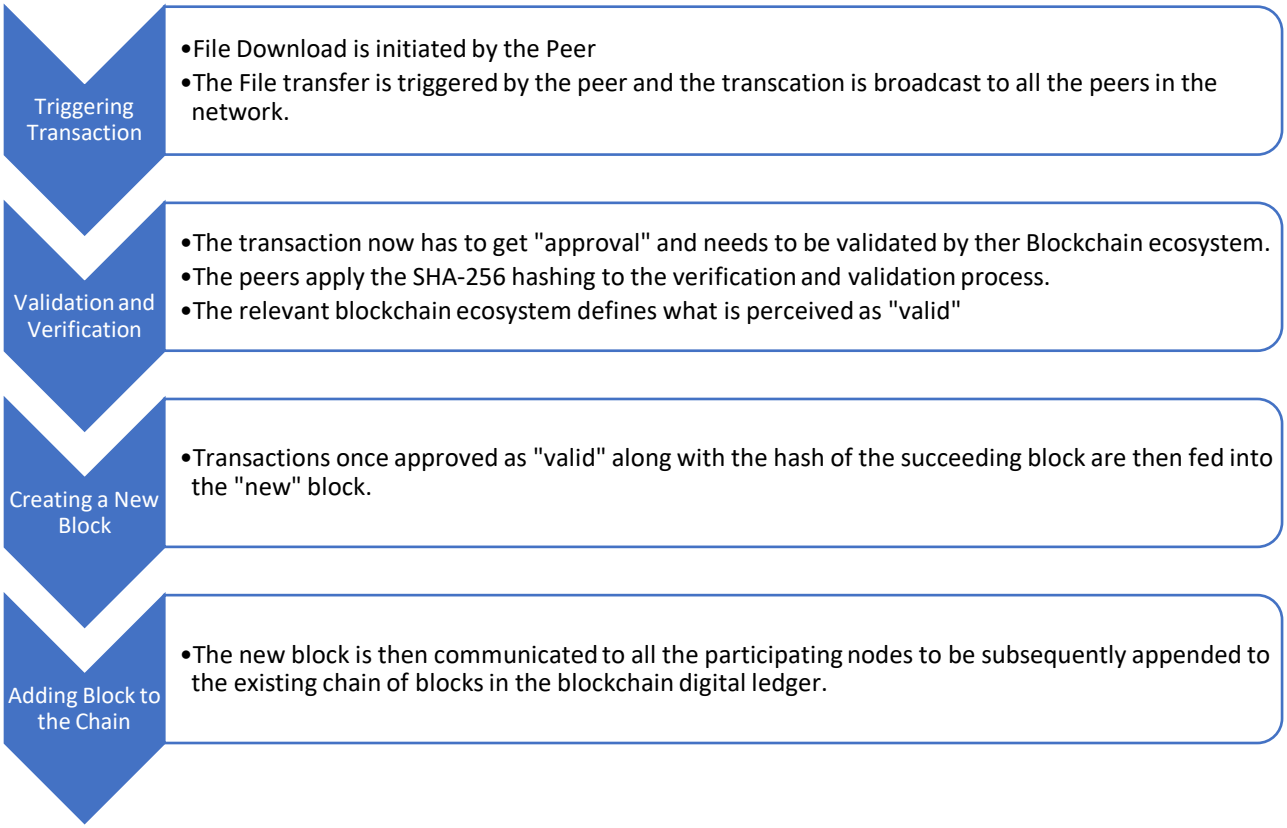


Fig 3: Flowchart depicting Blockchain Implementation procedure

PROCEDURE

A. Initialization

We used socket programming in Java to create a prototype of the peer to peer network. A control peer, acting as Distributed hash Table (DHT), connects all the peers all around the network to each other by creating an IP address list of all the peers and sharing the IP addresses with each system. Any peer who wants to participate in the network report to the control peer so that it gets added on to the distributed IP address list.

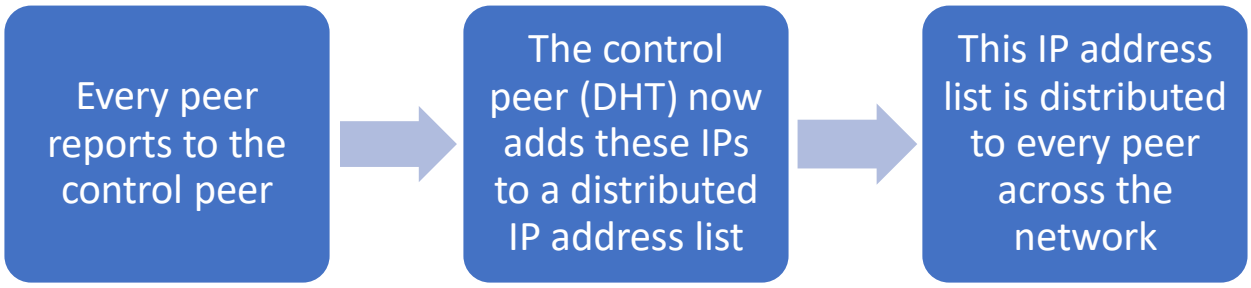


Fig 4: Flowchart to step-by-step Initialization

B. Uploading the file

When a peer wants to upload the data on the Decentralized cloud, it uses this list to determine the number of fragments the data should be broken down, with the use of erasure codes, such that it can calculate the number of peers to the number of fragments so that the network gains Pareto efficiency.

In order to ensure that correct packets go to the right nodes and transmission is secure (so that no user abuses our system to distributed malware), we use SHA checksum algorithm to check the integrity and security of the file. Since even very minor changes are depicted drastically in the SHA-256 checksum values we can ensure the integrity of the fragments.

This is done in the following way-

- Generating SHA-256 hash values for individual fragments and storing them as a model checksum.
- This SHA-256 hash is distributed as a secondary file as a ledger with the encrypted fragment.
- Now every node compares the original ledger to their version of generated SHA-256 hash.
- If the SHA-256 ledger matches with the real ledger values then the transaction is authenticated and this transaction is added to the blockchain.
- If it does not match then the transaction is rejected and the same encrypted fragment is requested from another node which has the fragment.

Now IP addresses are randomly selected from our IP address list and the encrypted fragments are transferred to those randomly selected systems. Which fragment goes to which system is saved in a file. A single fragment is sent over to many peers to ensure redundancy.

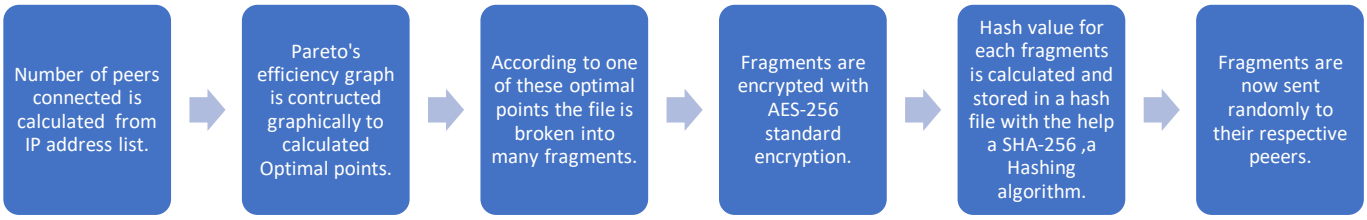


Fig 5: Flowchart to step-by-step File Upload to the Network

C. Retrieval of file from the network

Now when the peer wants to retrieve the data it sends requests to the DHT peer from that file which in turn requests all interacting peers to return the encrypted fragments. This request is added to the ledger of the blockchain. Before sending these fragments all peers are required to again use the SHA algorithm to generate Hash values and compare it with the distributed hash file.

Here is where the concept of blockchain is implemented. If each hash calculation matches for all peers all fragments get back to the user and this request is completed and added to the blockchain If any malicious user colludes and tries to send in a corrupted file to the user, then the hash values don't match and that transaction never occurs. Nobody but the user with the IP address list, the hash list and the decryption key can extract the file from the peer network unlike current systems where you have to trust the individual-based company not to leak your data, unlike here, where the participating peers cannot leak your data because the just have a fragment of data and they do not know who the other peers are. The highest grade of safety lies in anonymity.

After the transaction occurs all the fragments are individually decrypted and assimilated to get back the original uploaded file.

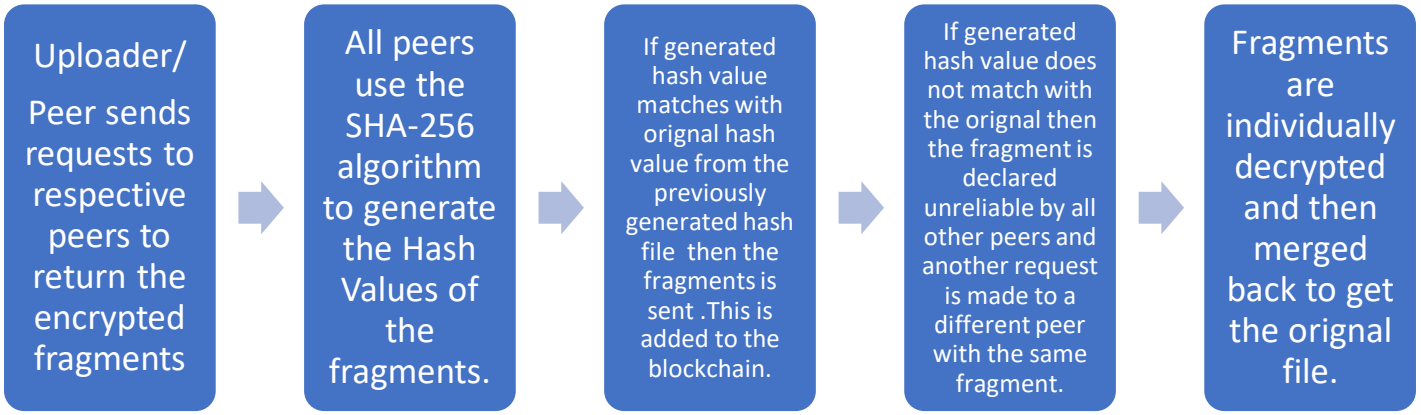
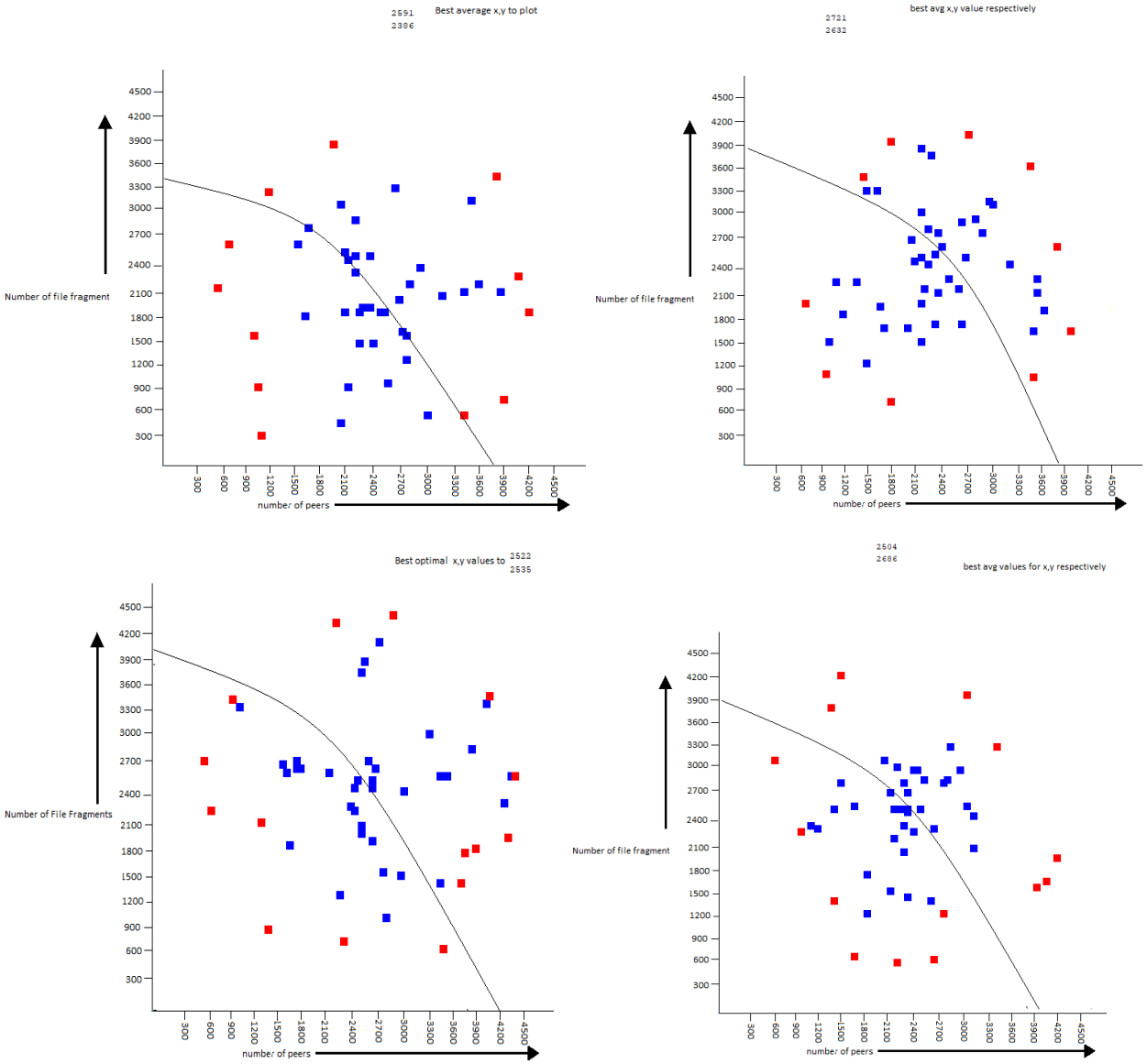


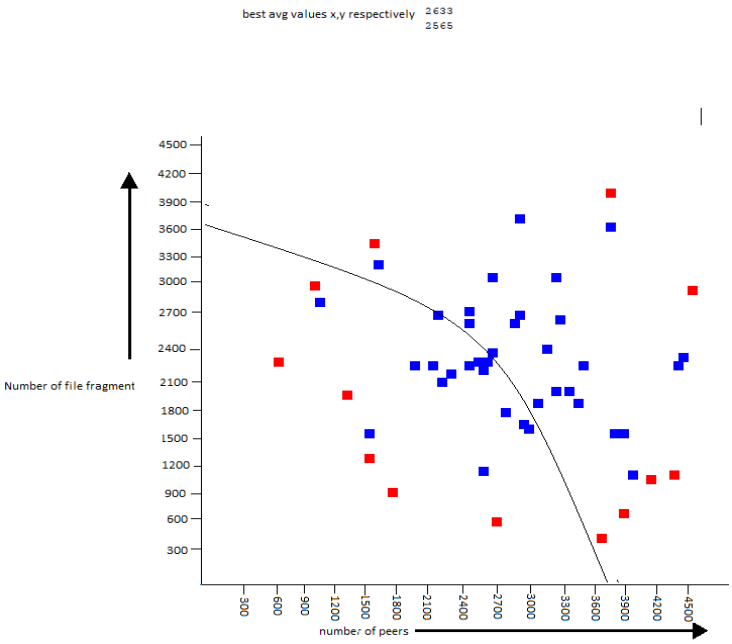
Fig 6: Flowchart to step-by-step retrieval of file from the network

RESULTS AND DISCUSSION

The Pareto efficiency graph is designed with a purpose of enabling optimal distribution so that an interdependent factor cannot be made more efficient without deteriorating the other factor. First, a Pareto set is created with respect to the number of peers connected and then after computation by the Pareto function, it is plotted on a graph.

In our prototype function, we add 50 elements to the Pareto's set and then it is graphed. The blue dots represent the optimal methods whereas the red dots represent the non-optimal methods. Now with respect to the blue dots, the best fit curve is drawn. All the points on or near to this best fit are almost Pareto efficient. We can utilize this data to make the network very efficient by making the fragment size optimal for that network.





These are the graphical representations of the pareto sets for a prototype network. Out of the many elements of that set, 50 random elements are chosen and plotted in each scatterplots.

Result:

5 Graphs are produced with each graph giving us comparably similar results, hence capable of predicting the Pareto's efficiency curve for the network.

Using this, we determine the fragment estimate of the file in direct determination with the number of peers available in our P2P Network.

Fig 7: Generated Graphical Representation of calculated Pareto Efficiency values using our code

	Predicted Optimal Number of Fragments(y)	Number of Peers Connected (x)	Probability Density (10^{-7})	Distribution Factor
Sample computation 1	2386	2591	1.4	0.999614
Sample computation 2	2633	2721	1.35	0.999632
Sample computation 3	2535	2522	1.57	0.999603
Sample computation 4	2686	2504	1.59	0.999600
Sample computation 5	2565	2633	1.44	0.999620

Fig 8: Table of Data estimated from our Program code generated graphical data

Findings

- The Distribution factor in our results is the fragment distribution accuracy so that the network becomes Pareto efficient. Since the fragment distribution accuracy tends to 1, we can conclude that our algorithm is successful at estimating the number of fragments to the number of peers.
- Probability density represents accuracy to the actual Pareto's constant so that the network almost tends to Pareto efficient.
- If Y is the exact value for Pareto efficiency, then deviation from $y = 1.4 \times 10^{-7}$
- The Distribution factor also tells us that the network can never be 100% Pareto efficient.

CONCLUSION

Personal data and sensitive data, in general, should not be trusted in the hands of third-parties, where they are susceptible to attacks and misuse. Instead, users should own and control their data without compromising security or limiting companies' and authorities' ability to provide personalized services. Our platform enables this by combining a blockchain, re-purposed as an access-control moderator, with a new blockchain solution. Users are not required to trust any third-party and are always aware of the data that is being collected about them and how it is used. In addition, the blockchain recognizes the users (peers) as the owners of their personal data.

- How such a peer to peer networking model helps us:

  P2P networks use a decentralized model in which each machine, referred to as a peer, functions as a client with its own layer of server functionality. A peer plays the role of a client and a server at the same time. That is, the peer can initiate requests to other peers, and at the same time respond to incoming requests from other peers on the network. It differs from the traditional client-server model where a client can only send requests to a server and then wait for the server's response.

- Performance of our P2P Network vs Cloud Server:

  With a traditional client-server approach, the performance of the server will deteriorate as the number of clients requesting services from the server increase. However, in P2P networks overall network performance actually improves as an increasing number of peers are added to the network. These peers can organize themselves into ad-hoc groups as they communicate, collaborate and share bandwidth with each other to complete the tasks at hand (e.g. file sharing). Each peer can upload and download at the same time, and in a process like this, new peers can join the group while old peers leave at any time. This dynamic re-organization of group peer members is transparent to end-users.

- Reliability and Security of our P2P Network

  Another characteristic of a P2P network is its capability in terms of fault-tolerance. When a peer goes down or is disconnected from the network, the P2P application will continue by using other peers, it is highly unlikely that all the peers would go down at once. Since the data

is stored on multiple peers randomly from the Distributed IP address list, malware cannot be targeted to a single system as in server-based cloud storage as it would be impossible to target the all the peers due to their sheer number and anonymity. Hence request overloading to stop the network from functioning is not possible in such a network.

- Multi-threaded Workloads:

  Smaller packet sizes and multiple peer sharing allows for multithreaded workloads to the network resulting in much faster transmission speeds, compared to uploading a whole file to a commercial server. By applying erasure codes, efficient packet transfer can be ensured as there is no ambiguity of packets getting lost or not during switching. Zero Packet loss results in no multiple requests for re-transmission of the packet, avoiding packet channel traffic and repetition in packets received. This makes for faster and more efficient data transfer over the network.

- **SAFE** Network- **S**ecure **A**ccess **F**or **E**veryone:

  This Blockchain Network is encrypted, autonomous, self-organizing as the blockchain consensus protocol needs all these decentralized nodes to reach agreement on a state within the network. Consensus occurs by each of these peers on the network checking the integrity of and safety of the transaction. If a malicious user tries to collude and distribute malware then his version of generated hash would never match the original one created at the time of uploading. Here all the peers would refuse such a transaction and our network would remain safe.

- Compartmentalization of sensitive data: *" Nobody spills the secrets because no one knows them all ".*

  Since the file as a whole is not stored on a particular system (like that of a server) but fragments of the file is stored on a group of peers randomly selected from our network, our data remains secure since that is just an encrypted fragment of the file. This crypto protocol renders a fragment of a data completely intelligible and useless as a means of extorting sensitive information. This makes it more secure than servers where all data is stored on a single system, making it a highly vulnerable target to be infiltrated by hackers. If an administrator of that server or the company is compromised and goes rogue, then that data can be leaked easily.

- Low Maintenance due to reduction of e-waste:

  This kind of an approach also reduces e-waste to a large extent as a major part of the e-waste belongs to the server and data processing industry and since the network does not require servers hence the need of upgrading servers once a year is eliminated. There is no need to index and repair the network regularly as Peer to peer-based system are self-healing.

The network is hence controlled by the whole community, not by a single peer. Even the control peer is not capable of damage as it only communicates the IP address list to the peers. A single malicious user can bring no harm to this network.

REFERENCES

[1] "Amazon's cloud crash disaster permanently destroyed many customers' data" April 28, 2011

http://www.thejournal.ie/amazons-cloud-crash-disaster-permanently-destroyed-many-customers-data-128127-Apr2011/

[2] "When Clouds Break: the Hidden Dangers of Cloud Computing" July 11, 2017

https://www.datacenterknowledge.com/industry-perspectives/when-clouds-break-hidden-dangers-cloud-computing

[3] "Amazon 'breaks the internet': massive server crash takes thousands of websites and apps from Slack to Soundcloud online" February 28, 2017

http://www.dailymail.co.uk/sciencetech/article-4268850/Amazons-cloud-service-partial-outage-affects-certain-websites.html

[4] "Facebook data leak whistleblower warns that your private info may be stored in Russia" April 9, 2018
https://bgr.com/2018/04/09/cambridge-analytica-facebook-scandal-private-data-russia/

[5] Pareto Efficiency from Wikipedia, the Free Encyclopedia **https://en.wikipedia.org/wiki/Pareto_efficiency**

[6] A note on data availability and erasure coding

https://github.com/ethereum/research/wiki/A-note-on-data-availability-and-erasure-coding

Other references-

[7] Muriel Medard and Steven S. Lumetta **"Network Reliability and Fault Tolerance"** 15 April, 2003

http://www.mit.edu/~medard/bchapter.pdf

[8] Wang Huan and Nakazato Hidenori **"Fault Tolerance in P2P- Grid Environments"**

2012 IEEE 26th International Parallel and Distributed Processing Symposium Workshops & Ph.D. Forum

https://ieeexplore.ieee.org/document/6270874/?part=1