PROJECT REPORT

On

Controlled Redundancy P2P Distributed Data Storage

Submitted for Partial Fulfillment of Award of

BACHELOR OF TECHNOLOGY

In

Computer Science and Engineering

(2023)

By

Rishabh Singh (Roll No: 1901220100090)

Rohan Sharma (Roll No: 1901220100092)

Under the Guidance

Of

Dr. Pankaj Kumar

SHRI RAMSWAROOP MEMORIAL COLLEGE OF ENGINEERING & MANAGEMENT, LUCKNOW

Affiliated to

Dr. APJ ABDUL KALAM TECHNICAL UNIVERSITY, LUCKNOW

DEPARTMENT OF
COMPUTER SCIENCE AND ENGINEERING

SRMCEM

CERTIFICATE

Certified that the project entitled "**Controlled Redundancy P2P Distributed Data Storage**" submitted by **Rishabh Singh [Unv. Roll No. 1901220100090]** and **Rohan Sharma [Unv. Roll No. 1901220100092]** in the partial fulfillment of the requirements for the award of the degree of Bachelor of Technology (Computer Science and Engineering) of Dr. APJ Abdul Kalam Technical University (Lucknow), is a record of student's own work carried under our supervision and guidance. The project report embodies the results of original work and studies carried out by students and the contents do not forms the basis for the award of any other degree to the candidate or to anybody else.

Dr. Pankaj Kumar
Associate Professor, Head of Department
Computer Science and Engineering
(Project Guide and Head of Department)

DEPARTMENT OF
COMPUTER SCIENCE AND ENGINEERING

SRMCEM

DECLARATION

We hereby declare that the project entitled "**Controlled Redundancy P2P Distributed Data Storage**" submitted by me/us in the partial fulfillment of the requirements for the award of the degree of Bachelor of Technology (Computer Science and Engineering) of Dr. APJ Abdul Kalam Technical University (Lucknow), is record of my/our own work carried under the supervision and guidance of **Dr. Pankaj Kumar** Head of Department.

To the best of my/our knowledge, this project has not been submitted to Dr. APJ Abdul Kalam Technical University (Lucknow) or any other University or Institute for the award of any degree.

Rishabh Singh

Unv. Roll No. 1901220100090

Rohan Sharma

Unv. Roll No. 1901220100092

ACKNOWLEDGEMENT

We are deeply indebted to our Head of Department **Dr. Pankaj Kumar** (HOD), Department of Computer Science and Engineering, Shri Ramswaroop Memorial College Of Engineering And Management, who modeled us both technically and morally for achieving greater success in life. We would like to express our gratitude to Dr. V. K. Singh, Director SRMCEM, for his support which gives us encouragement to complete our work.

Also, we would like to express our sincere thanks to our project guide **Dr. Pankaj Kumar** (HOD) for his constant encouragement and support given during the course of the project period.

We would thank all the faculty of our college for their help and assistance in making this project a successful one. Finally, we take this opportunity to extend our deep appreciation to our parents and family and all friends who were a great source of inspiration to us and for all that they meant to us during the crucial times of the completion of our project.

Rishabh Singh

University Roll No. 1901220100090

SRMCEM

Rohan Sharma

University Roll No. 1901220100092

SRMCEM

PREFACE

The Design and Development of a, "Controlled Redundancy P2P Distributed Data Storage", new network will run on the existing torrent network owing to its widespread popularity and proven effectiveness. However, it will implement an additional layer on top of BitTorrent which will allow for the tracking of the redundancy rate of individual files and controlling them.

INTRODUCTION - In this chapter, we have discussed the basic concept of the project.

LITERATURE REVIEW - In this chapter, we have discussed about all the research papers we have read and analyzed to accomplish the proposed project.

PROPOSED METHODOLOGY -In this chapter, we have discussed about the Proposed Methodology and Experimental setup technology.

RESULTS ANALYSIS AND DISCUSSION - In this chapter we have discussed about all results noted and discussion on it.

CONCLUSION - In this chapter, we have discussed about the conclusion of our project.

FUTURE SCOPE – In this chapter, we have discussed about the Future Scope of our project.

ABSTRACT

The efficient and optimized distribution of files in peer-to-peer networks is a critical aspect of torrenting protocols. We propose a novel solution that builds on top of existing torrenting protocols to track the redundancy rate of files in a network and optimize file downloads. Our proposed network leverages redundancy tracking to instruct nodes to download files that are below a predefined redundancy target, thereby reducing unnecessary duplication of files and optimizing bandwidth usage. Additionally, our solution provides information to nodes about heavily seeded files, allowing them to reclaim disk space by deleting redundant files. We present a comprehensive system architecture and implementation details of our proposed solution, along with an evaluation of its performance through experiments or simulations. The results show that our solution improves the efficiency of file distribution in torrenting networks and has the potential to significantly reduce bandwidth usage and storage requirements. The findings of this research contribute to the field of peer-to-peer networks and offer insights for further research and development in optimizing torrenting protocols.

Keywords - torrenting protocols, redundancy tracking, file downloads, peer-to-peer networks, efficiency, optimization, redundancy rate, bandwidth usage, disk space reclamation, file distribution, system architecture, implementation details, evaluation, performance, research, peer-to-peer networks

TABLE OF CONTENT

Chapter-1: Introduction

Peer-to-peer (P2P) file sharing is a widely adopted method for distributing large files over the internet. However, the issue of file redundancy in P2P networks poses challenges in terms of network efficiency and resource utilization [1]. Various approaches have been proposed to address this problem, including file popularity metrics, distributed hash tables (DHTs), and peer cooperation mechanisms [2][3][4].

We propose a novel solution that builds upon existing torrenting protocols to tackle the issue of file redundancy in P2P file sharing networks. Our solution tracks the redundancy rate of individual files and leverages this information to optimize file distribution and storage allocation [7][10]. By instructing nodes to download files below a specified redundancy target value, we ensure efficient utilization of network bandwidth and storage resources [5]. Additionally, our solution enables nodes to identify heavily seeded files, allowing for their deletion to reclaim disk space [9].

The proposed network architecture incorporates several key modules. The beacon module serves as the primary link in the beacon chain, responsible for maintaining the random beacon chain for torrents and synchronizing it with peer nodes through peer-to-peer communication [1]. The node module facilitates the discovery and connection of nodes within the network, ensuring reliable and efficient routing of requests [2]. The user dashboard provides users with a comprehensive overview of their network and torrent status, enabling effective management of shared files [4]. The torrent module encompasses the necessary libraries for interacting with the BitTorrent architecture, enabling seamless data sharing [6]. The database module allows users to submit requests by specifying the required items, enhancing the usability and versatility of the network [8]. Finally, the create torrent module empowers users to add their desired torrent details to be shared within the network [9].

Our proposed solution offers several advantages over existing methods. It seamlessly integrates with established torrenting protocols, ensuring compatibility and ease of implementation [10]. By optimizing file distribution based on redundancy rates, it enhances network efficiency and reduces

unnecessary file transfers [5]. Moreover, the ability to reclaim disk space by identifying heavily seeded files alleviates storage constraints and improves overall system performance.

In the subsequent sections, we delve into the detailed implementation and system architecture of our proposed solution. We also discuss future prospects and potential research directions to further enhance the efficiency and effectiveness of P2P file sharing networks.

A distributed data storage system would make data almost invulnerable to being lost or deleted as is possible in the current centralized model of the Internet. The current solution to this is torrenting. While popular and hugely successful, torrenting has a major disadvantage; the users willing to participate in it need to download and seed a huge amount of data.

We propose a possible solution with a network built on top of existing torrenting protocols. The network tracks the redundancy rate of each file in a network and consequently instructs the nodes to download files that are below the redundancy target value. It can also inform the nodes which files are heavily seeded and thus allow them to delete the files and reclaim disk space.

Fig. 1.1: Sharing of data

Fig. 1.2: Wastage of Storage drives

1.1 Problem Definition

1. Maintaining the target redundancy rate for each file throughout the network regardless of file size and peer count.

2. Compensating for a low redundancy rate by starting file seeding throughout the network by using regular polling.

3. Decreasing network load on individual peers and reducing memory footprint on every node that partakes in the network.

4. Combating byzantine nodes and having multiple layers of security in beacon nodes.

1.2 Project Objectives

1. The project aims to solve the problem of participating in a P2P network serving large-sized files.

2. Our implementation aims to replicate only some files to some clients which will ensure that no one peer has to burden itself with downloading and seeding and the high memory requirement that comes with it.

3. It also aims to create a fault-tolerant network wherein a number of peers/nodes going down would not affect the working nodes and the network would adjust to the current network condition.

Chapter-2: Literature Survey

Peer-to-peer (P2P) file sharing networks have been extensively studied, and several approaches have been proposed to address the challenges associated with file redundancy and network efficiency. These studies provide valuable insights into the existing solutions and form the foundation for our proposed approach.

Cohen's work on BitTorrent [1]: Cohen's seminal work on BitTorrent emphasized the importance of incentives in building robustness within P2P networks. He introduced the concept of tit-for-tat mechanisms, where peers share files with those who reciprocate sharing. This approach promotes cooperation and enhances the efficiency of file distribution. Choffnes and Bustamante on reducing cross-ISP traffic in P2P systems [2]: This study focused on practical approaches to minimize inter-ISP communication and improve network performance. The researchers explored techniques such as localizing peer discovery and favoring local peer connections, which reduced the load on ISPs and enhanced network efficiency.

Cohen's work on BitTorrent highlighted the importance of incentives in building robustness within the network [1]. The study emphasized the role of tit-for-tat mechanisms and peer cooperation in enhancing the efficiency of file distribution. Choffnes and Bustamante focused on reducing cross-ISP traffic in P2P systems, proposing practical approaches to tame the torrent and improve network performance [2]. They explored techniques such as localizing peer discovery and favoring local peer connections to minimize inter-ISP communication.

Dinger and Kolberg's survey of BitTorrent enhancements [3]: This survey covered various aspects of BitTorrent, including distributed tracking mechanisms, piece selection strategies, and optimization techniques. It provided an overview of existing improvements to BitTorrent and their impact on efficiency and robustness.

Dinger and Kolberg conducted a survey of BitTorrent enhancements aimed at improving efficiency and robustness [3]. The survey covered various aspects, including distributed tracking mechanisms, piece selection strategies, and optimization techniques. Gupta et al. conducted an empirical analysis of BitTorrent's incentive mechanisms, shedding light on the effectiveness and impact of different

incentive models on network behavior [4]. Their findings provided valuable insights into the dynamics of peer interactions and the implications for file sharing efficiency.

Gupta et al.'s empirical analysis of BitTorrent's incentive mechanisms [4]: This study analyzed the effectiveness and impact of different incentive models in BitTorrent. By examining the dynamics of peer interactions, the researchers provided insights into the implications of incentive mechanisms on network behavior and file sharing efficiency. This research explored the integration of reputation-based mechanisms to incentivize cooperation among peers in BitTorrent. The study demonstrated the potential of reputation systems in enhancing file distribution and improving network efficiency.

Reputation-based incentives have also been investigated as a means to improve BitTorrent's performance. Reindl, Hotho, and Stumme explored the integration of reputation-based mechanisms to enhance file distribution and incentivize cooperation among peers [5]. Their work demonstrated the potential of reputation systems in improving the overall network efficiency.

Analysis of YouTube network traffic

This study focused on the challenges and implications of video content distribution within a campus network. By measuring and analyzing YouTube network traffic, the researchers provided insights into the characteristics of video distribution in P2P networks.

Other studies have focused on specific aspects of P2P file sharing networks. Zink et al. conducted measurements and analysis of YouTube network traffic, highlighting the challenges and implications of video content distribution within a campus network [6]. Halkes and Pouwelse examined seed replication in BitTorrent, evaluating the performance and incentives associated with replicating popular files to enhance availability and download speeds [7]. This study examined the performance and incentives associated with seed replication in BitTorrent. The researchers evaluated the benefits of replicating popular files to enhance availability and download speeds, contributing to improved network efficiency.

Remaining Literature Survey:

Ruotsalainen and Gurtov evaluated different file location mechanisms based on distributed hash tables (DHTs) in BitTorrent networks, providing insights into the effectiveness of these mechanisms in facilitating efficient resource discovery [8]. This research focused on different file location mechanisms based on distributed hash tables (DHTs) in BitTorrent networks. The study provided insights into the effectiveness of these mechanisms in facilitating efficient resource discovery.

Xiong, Yu, and Liu proposed techniques for tracking torrent downloaders, focusing on filtering and discovery mechanisms to improve the efficiency and accuracy of tracking in P2P networks [9]. This study focused on filtering and discovery mechanisms to improve the efficiency and accuracy of tracking in P2P networks. The researchers proposed techniques for tracking torrent downloaders, contributing to better network management and resource allocation.

While the existing literature has made significant contributions to understanding and enhancing P2P file sharing networks, there is still room for innovation and improvement. In the subsequent sections, we present our proposed solution that leverages the insights from these studies and addresses the challenges of file redundancy and network efficiency in a novel manner.

2.1 Understanding the need for controlled redundancy

Understanding the need for controlled redundancy is essential in the context of optimizing database access and improving query performance. Redundancy, in the traditional sense, refers to storing the same data or information multiple times in a database, which can lead to various issues such as duplication of effort, wastage of storage space, and inconsistent data.

Controlled redundancy, on the other hand, is a technique that selectively replicates certain data fields in a database to enhance access speed and query performance. By strategically duplicating stable and infrequently changing data from a parent entity to a child entity, controlled redundancy reduces the number of database accesses required for read operations.

In the given scenario, the problem revolves around the need to efficiently handle test invoices and improve the speed of reading data from a parent entity, particularly the Order table. Despite denormalizing the table, the query for printing orders still incurs heavy database load due to the multiple accesses required for order positions, customer data, and article data.

To address this issue, the solution suggests implementing controlled redundancy. By replicating relevant data from the parent entity (Article) into the child entity (OrderItem), the number of database pages accessed for read operations can be significantly reduced. It is crucial to replicate only stable data that are not subject to frequent updates, ensuring that the redundancy remains consistent.

However, implementing controlled redundancy comes with certain consequences. While it improves read performance, it may require additional database accesses for write operations on the parent entity. Additionally, controlled redundancy increases the database space needed to accommodate redundant data. The code complexity also increases as application-level considerations are brought down to the database level. It is important to shield and control controlled redundancy through a Physical Access Layer to prevent code clutter and maintain a clear separation of concerns.

It is worth noting that redundancy does not impair the understandability of the physical data model; in fact, it can potentially enhance it. The decision to implement controlled redundancy should be based on the stability of the data and the tradeoff between read and update performance. Understanding the need for controlled redundancy arises from the aim to optimize database access, reduce the number of database pages accessed for read operations, and improve query performance. It involves selectively replicating stable data in a controlled manner, considering factors such as data stability, space requirements, code complexity, and query performance.

2.1.1 Existing Implementation 01 - Bittorrent

Fig. 2.1: Bittorrent

In this section we will be taking a look at existing technologies and comparing them to our own.

Current Implementations:

The most popular technology currently being used for P2P file transfer is Bittorrent. It allows for seeding and downloading files without the need for a centralized server. This is very effective however it suffers from the aforementioned problems of high bandwidth and memory requirements.

2.1.2 Existing Implementation 02 - Cassandra DB

Fig. 2.2: Cassandra DB

There also exist some redundancy-controlled storage solutions but they either lack P2P or have different use cases, like databases. Cassandra DB

Current Implementations:

It is a distributed, wide-column store, NoSQL database management system designed to handle large amounts of data across many commodity servers, providing high availability with no single point of failure. Cassandra offers support for clusters spanning multiple data centers, with asynchronous masterless replication allowing low-latency operations for all clients.

2.2 Survey Conclusion

The literature survey reveals the extensive research and development efforts in the field of peer-to-peer (P2P) file sharing networks. The studies discussed various aspects, including network efficiency, file distribution, incentive mechanisms, and redundancy control. Cohen's work on BitTorrent highlighted the importance of incentives and peer cooperation, while Choffnes and Bustamante focused on reducing cross-ISP traffic. Other studies explored enhancements, reputation-based incentives, video content distribution challenges, seed replication, file location mechanisms, and torrent tracking techniques.

While the existing literature has provided valuable insights and proposed effective solutions, there is still room for innovation and improvement in P2P file sharing networks.

Chapter-3: Proposed Methodology

The new network will run on the existing torrent network owing to its widespread popularity and proven effectiveness.

However, it will implement an additional layer on top of BitTorrent which will allow for the tracking of the redundancy rate of individual files and controlling them.

For maximum performance and hole punching (for Network Translation Layer) User Datagram Protocol will be used.

Python programming language will be used for simplicity and rich libraries.

For minimal overhead costs, low level python libraries will be used such as socket and io.

3.1 Proposed Solution

The proposed solution builds on top of existing BitTorrent protocols to create an additional layer that allows for tracking the redundancy rate of individual files and controlling them. This solution consists of several modules that work together to optimize file sharing in the torrent network.

1. **Beacon Module**: The beacon module acts as the primary link in the beacon chain, which forms the backbone of the torrent network. It is responsible for running and maintaining the random beacon chain for the torrents. The beacon module also syncs the beacon network with peer nodes via peer-to-peer communication, ensuring consistency and accuracy in tracking the redundancy rate of files.

2. **Node Module**: The node module manages the nodes in the network, ensuring that requests always find the destination node that holds the requested resource. This module handles node joining and leaving events, maintaining an updated list of active nodes in the network, and routing requests to the appropriate nodes.

3. **User Dashboard**: The user dashboard provides a comprehensive overview of the user's network and torrent status. Once the user has successfully generated a torrent, they can access the dashboard to view the status of their torrents, including information on redundancy rate, availability, and download progress.

4. **Torrent Module**: The torrent module contains the necessary libraries for communicating with the BitTorrent architecture for handling data sharing. This module implements the proposed solution's logic for tracking the redundancy rate of files and optimizing file downloads based on the redundancy target value.

5. **Database Module**: The database module allows users to make requests by adding the details of the required item. It maintains a database of files available in the network and their redundancy rates, which is used by the torrent module for making informed decisions on file downloads.

6. **Create Torrent Module**: The create torrent module allows users to add their torrent details, including the files they want to share in the network. This module interfaces with the torrent module to generate torrents with appropriate redundancy targets based on the network's current redundancy rate.

The proposed solution integrates these modules to create an efficient and scalable approach for tracking the redundancy rate of files, optimizing file downloads, and controlling redundancy in the torrent network. By leveraging the existing BitTorrent protocols and adding an additional layer, this solution aims to improve the efficiency and effectiveness of file sharing in the torrent network while minimizing duplication and optimizing disk space utilization.

3.2 System Architecture

The proposed solution for the network built on top of existing torrenting protocols consists of several modules that work together to optimize file sharing. The system architecture is designed to track the redundancy rate of files and control them, while leveraging the proven effectiveness of the BitTorrent protocol.

The main components of the system architecture include:

1. **Beacon Module**: The beacon module is responsible for running and maintaining the random beacon chain for the torrents. It syncs the beacon chain with peer nodes via peer-to-peer communication, ensuring consistency in tracking the redundancy rate of files across the network. The beacon module acts as the primary link in the beacon chain, forming the backbone of the system.

2. **Node Module**: The node module manages the nodes in the network and handles node joining and leaving events. It maintains an updated list of active nodes in the network and routes requests to the appropriate nodes. The node module ensures that requests always find the destination node that holds the requested resource, enabling efficient file sharing.

3. **User Dashboard**: The user dashboard provides a comprehensive overview of the user's network and torrent status. It allows users to view the status of their torrents, including information on redundancy rate, availability, and download progress. The user dashboard serves as a user-friendly interface for managing torrents and monitoring their performance.

4. **Torrent Module**: The torrent module contains the necessary libraries for communicating with the BitTorrent architecture and implementing the proposed solution's logic for tracking the redundancy rate of files and optimizing file downloads. It interfaces with the beacon module to obtain information on the redundancy rate of files and makes informed decisions on file downloads based on the redundancy target value.

5. **Database Module**: The database module maintains a database of files available in the network and their redundancy rates. It stores information on the redundancy rate of files as reported by the beacon module. The database module is used by the torrent module for making decisions on file downloads, ensuring that files below the redundancy target value are prioritized.

The system architecture is designed to be scalable and efficient, leveraging the existing BitTorrent protocols and adding an additional layer for tracking redundancy rate and controlling files in the network. The various modules work together to optimize file sharing, improve network efficiency, and enhance user experience in the torrent network.

3.3 SOFTWARE AND HARDWARE REQUIREMENTS

3.3.1 SOFTWARE REQUIREMENTS

- **Front End** : HTML5, CSS, BootStrap, JS
- **Runtime Environment** : Node JS
- **Backend Server** : Express JS, React JS
- **Database Tool** : MongoDB, SQL
- **IDE** : Visual Studio Code

3.3.2 HARDWARE REQUIREMENTS

- **PROCESSOR** **:** Intel core i3 or Above
- **RAM** **:** 2 GB and above
- **HDD** **:** 512 MB
- **O.S** : Windows 8.1 or higher
- **Screen** : VGA Monitor or Laptop Screen Or Mobile
- **Network** **:** Internet, or any Network
- **Keyboard** **:** Standard Keyboard (QWERTY)

3.4 Technologies Used

The implementation involves the use of the following technologies:

Fig. 3.1: HTML icon

Fig. 3.2: CSS icon

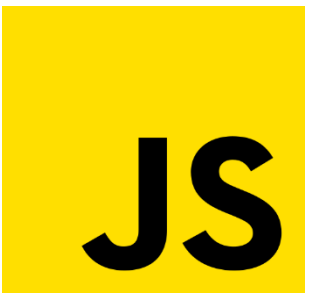


Fig. 3.3: JavaScript icon

Fig. 3.4: ExpressJS icon

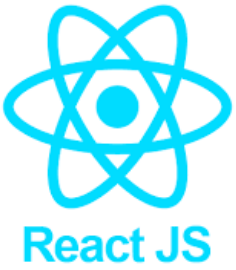


Fig. 3.8: React.js icon

Fig. 3.5: NodeJS icon

Fig. 3.6: MongoDB icon

Fig. 3.7: BootStrap icon

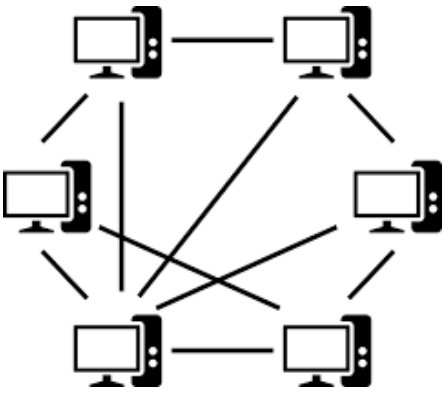


Fig. 3.9: Peer to peer network

3.4.1 HTML5

HTML5 is a markup language used for structuring and presenting content on the World Wide Web. The fifth and last major HTML version is a World Wide Web Consortium (W3C) recommendation. The current specification is known as the HTML Living Standard. HTML5 includes detailed processing models to encourage more interoperable implementations; it extends, improves, and rationalizes the markup available for documents and introduces markup and application programming interfaces (APIs) for complex web applications. For the same reasons, HTML5 is also a candidate for cross-platform mobile applications because it includes features designed with low-powered devices in mind.

3.4.2 CSS3

CSS3 is the latest version of the CSS specification. CSS3 adds several new styling features and improvements to enhance the web presentation capabilities. CSS gives lots of flexibility to set the style properties of an element. You can write CSS once, and then the same code can be applied to the groups of HTML elements and can also be reused in multiple HTML pages.CSS provides an easy means to update the formatting of the documents, and to maintain consistency across multiple documents. Because the content of the entire set of web pages can be easily controlled using one or more style sheets.

3.4.3 JAVASCRIPT

JavaScript often abbreviated as JS, is a programming language that conforms to the ECMAScript specification. JavaScript is high-level, often just-in-time compiled, and multi-paradigm. It has dynamic typing, prototype-based object orientation, and first-class functions. All major web browsers have a dedicated JavaScript engine to execute the code on the user's device. As a multi-paradigm language, JavaScript supports event-driven, functional, and imperative programming styles. It has application programming interfaces (APIs) for working with text, dates,regular expressions, standard data structures, and the Document Object Model (DOM).

3.4.4 EXPRESS JS

Express.js, or simply Express, is a back-end web application framework for Node.js, released as free and open-source software under the MIT License. It is designed for building web applications and APIs. It has been called the de facto standard server framework for Node.js.The original author, TJ Halewyck, described it as a Sinatra-inspired server, meaning, that it is relatively minimal with many features available as plugins. Express is the back-end component of popular development stacks like the MEAN, MERN or MEVN stack, together with the MongoDB database software and a JavaScript front-end framework or library.

3.4.5 NODE JS

Node.js is an open-source, cross-platform, back-end JavaScript runtime environment that runs on the V8 engine and executes Javascript code outside a web browser. Node.js lets developers use JavaScript to write command line tools and for server-side scripting-running scripts server-side to produce dynamic web page content before the page is sent to the user's web browser. Consequently, the Node.js paradigm, unifying web applications represents a "JavaScript everywhere" development around a single programming language, rather than different languages for server-side and client-side scripts.

3.4.6 MONGO DB

MongoDB is an open-source document database and leading NoSQL database. MongoDB is written in C++. This tutorial will give you a great understanding of MongoDB concepts needed to create and deploy a highly scalable and performance-oriented database.
MongoDB is a source-available cross-platform document-oriented database program. Classified as a NoSQL database program, MongoDB uses JSON-like documents with optional schemas. MongoDB is developed by MongoDB Inc. and licensed under the Server-Side Public License (SSPL).

3.4.7 BOOTSTRAP

Bootstrap is a free and open-source CSS framework directed at responsive, mobile-first front-end web development. It contains CSS- and (optionally) JavaScript-based design templates for typography, forms, buttons, navigation, and other interface components. The primary purpose of adding it to a web project is to apply Bootstrap's choices of color, size, font, and layout to that project. As such, the primary factor is whether the, developers in charge find those choices to their liking. Once added to a project, Bootstrap provides basic style definitions for all HTML elements. The. In addition, developers can take advantage of CSS classes defined in Bootstrap to further customize the appearance of their content.

3.4.8 REACT JS

React is a free and open-source front-end JavaScript library for building user interfaces based on UI components. It is maintained by Meta and a community of individual developers and companies.
The React. js framework and library developed by Facebook. It's used for building interactive user interfaces and web applications quickly and efficiently with significantly less code than you would with vanilla JavaScript.

3.4.8 P2P

In a peer-to-peer (P2P) network, each computer acts as both a server and a client—supplying and receiving files—with bandwidth and processing distributed among all members of the network. Such a decentralized network uses resources more efficiently than a traditional network and is less vulnerable to systemic failure.

3.5 IMPLEMENTATION DETAILS

The proposed solution for the network built on top of existing torrenting protocols involves the use of Python for the backend and ReactJS for the frontend. The backend is responsible for handling the beacon module, which uses flooding to discover and connect with other beacons in the network. The discovered beacons then share file statistics and collectively decide on which files to download or delete based on redundancy rates

1. **Backend Implementation**: The backend is implemented in Python, which is a versatile and widely used programming language known for its ease of use and robustness. The backend handles the beacon module, which is responsible for running and maintaining the random beacon chain for the torrents. The beacon module uses flooding, a distributed network discovery technique, to discover and connect with other beacons in the network. The beacon module exchanges file statistics with other beacons, including redundancy rates, to collectively decide on which files to download or delete.

2. **Frontend Implementation**: The frontend is implemented using ReactJS, a popular JavaScript library for building user interfaces. ReactJS provides a robust and efficient way to create interactive and responsive web interfaces, making it suitable for implementing the user dashboard and other frontend components of the proposed solution. The user dashboard allows users to view the status of their torrents, including file statistics such as redundancy rates, availability, and download progress. Users can also create torrents with appropriate redundancy settings using the create torrent module, which interfaces with the backend to generate torrents with optimal redundancy targets.

3. **Flooding for Beacon Discovery**: The beacon module uses flooding as a network discovery technique to discover and connect with other beacons in the network. Flooding involves broadcasting beacon messages to all neighboring nodes, which in turn rebroadcast the messages to their neighbors until all nodes in the network receive the beacon messages. This

allows beacons to discover each other and form connections for exchanging file statistics and collectively deciding on file downloads or deletions.

4. **File Statistics Exchange**: Once the beacons have discovered each other through flooding, they exchange file statistics, including redundancy rates, to collectively decide on which files to download or delete. The beacon module uses the received file statistics to update the local database module, which maintains information on the redundancy rate of files available in the network. This allows the torrent module to make informed decisions on file downloads based on the redundancy target value, optimizing file sharing in the network.

5. **Redundancy-based File Download and Deletion**: Based on the file statistics received from other beacons, the torrent module in the backend determines which files are below the redundancy target value and need to be downloaded to maintain redundancy. It instructs the nodes in the network to download those files to ensure optimal redundancy levels. Similarly, the torrent module also determines which files are heavily seeded and can be deleted to reclaim disk space. The backend communicates with the frontend to update the user dashboard with the status of files, including download progress and deletion status.

The implementation details of the proposed solution involve the use of Python for the backend, ReactJS for the frontend, flooding for beacon discovery, and file statistics exchange for collective decision-making on file downloads and deletions. The backend and frontend components work together to optimize file sharing, improve network efficiency, and enhance user experience in the network.

3.6 MODULE DESCRIPTION

3.6.1. Beacon Module

The beacon node is the primary link in the beacon chain that forms the backbone of the torrent network. Some of the beacon node's responsibilities are:

1. Run and maintain the random beacon chain for the torrents.
2. Sync this beacon chain with peer nodes via peer to peer.

3.6.2. Node Module

Nodes come and leave the network. To ensure that requests always find the destination node (the one that holds the requested resource).

3.6.3 User Dashboard

Once the user has successfully generated the torrent, fill out all the details on the creation page. He'll be able to access the dashboard which will show a complete overview of his network and torrent’s status.

3.6.4 Torrent Module

This homepage of the portal consists of a simple UI which has info about requested items, matched items and has a feature to add new request, remove old request, add new item or remove previously shared item.

3.6.5 Database Module

Users can use this feature to make requests by adding the details of the required item.

3.6.6 Create Torrent Module

Users can add their item details which they want to sell/share.

Chapter-4: Result Analysis and Discussion

4.1 Load analysis

A good way of understanding the need for such a network would be to look at the difference in loads on peers using the traditional torrent technologies and a controlled redundancy network.

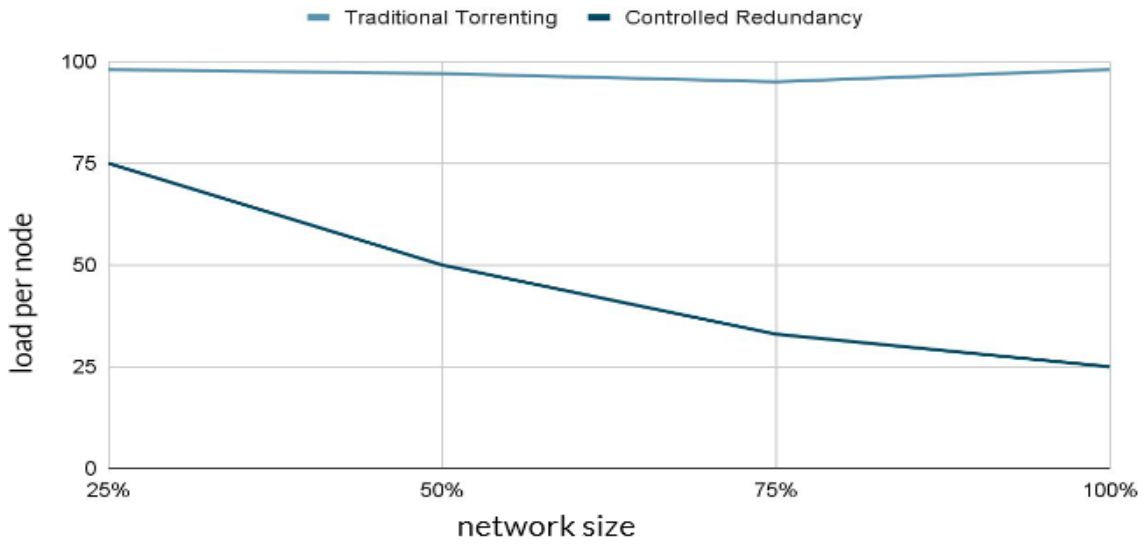


Fig. 4.1: Load analysis

Fig. 4.2: Load per peer

The reduction of network load with increasing peers means that as a network grows, more peers can take part in it which reduces the load even further. This incentivizes more people to join and thus forms a feedback loop.

Since the load on every peer is lowered, a lesser cost is required for setting up the necessary network and storage hardware.

4.2 Target audience

The project's objective is to develop a product that caters to a diverse range of users with different needs and motivations. The target audience for this product includes:

Large Enterprises that need to store public data:
Large enterprises often deal with a vast amount of public data, including user-generated content, documents, media files, and more. They require robust and scalable storage solutions that can handle their data storage needs efficiently. The proposed product aims to provide a reliable and redundant storage system that can meet the requirements of these enterprises, ensuring data integrity and accessibility.

Regular Users who want to contribute to securing data:
There are individuals who are concerned about data security and want to actively contribute to safeguarding public data. These users may have personal or professional interests in preserving important information, historical records, or cultural artifacts. The product targets these users by providing them with a platform where they can securely store and contribute to the preservation of valuable data.

Large-scale Databases like Wikipedia/Z-Library:
Large-scale databases, such as Wikipedia and Z-Library, are repositories of vast amounts of information accessible to the public. These databases rely on a decentralized and redundant infrastructure to ensure data availability and prevent data loss. The proposed product can offer a reliable and distributed storage solution to support these databases, enhancing their data resilience and availability.

People who want to circumvent national censorship:
In some regions, internet censorship and restrictions on accessing certain types of information are prevalent. There are individuals who desire to bypass these restrictions and access content freely. The proposed product can provide a means for these users to securely store and share

information without being subjected to censorship, ensuring data availability even in restricted environments.

Data Hoarders:

Data hoarders are individuals who collect and preserve large amounts of data, often spanning various topics and formats. They are motivated by the desire to archive and preserve information that might otherwise be lost or forgotten. The product can cater to data hoarders by offering a reliable and scalable storage solution that allows them to store and manage their vast collections of data efficiently.

Overall, the target audience of this product encompasses large enterprises in need of secure and scalable data storage, regular users interested in contributing to data security, large-scale databases requiring redundant infrastructure, individuals seeking to circumvent censorship, and data hoarders who want efficient storage for their extensive collections. By addressing the diverse needs of these user groups, the proposed product aims to provide a comprehensive solution for secure and reliable data storage.

4.3 Screenshot and Diagram

Block Diagram

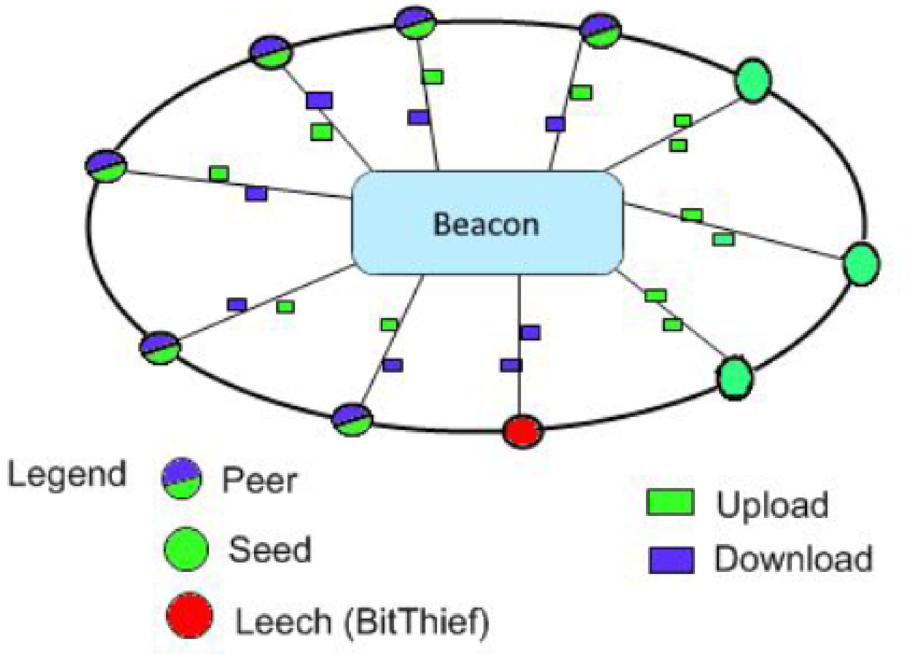


Fig. 4.3: Block Diagram

USE CASE DIAGRAM

Files in each collection:

Collection 1:
1. C1 File 1
2. C1 File 2
3. C1 File 3
4. C1 File 4
5. C1 File 5

Collection 2:
1. C2 File 1
2. C2 File 2

Starting Distribution (Initial distribution of files, not available or not reliable)

Peer	Peer	Peer	Peer	Peer	Peer	Peer	Peer
C1 File 1		C1 File 1	C2 File 1	C1 File 2	C1 File 1	C1 File 2	C1 File 1
C1 File 2		C1 File 3	C2 File 2	C2 File 2	C1 File 2	C2 File 2	
C2 File 1		C1 File 4					
		C1 File 5					

Existing Solution (Needs to download and store all the files)

Peer	Peer	Peer	Peer	Peer	Peer	Peer	Peer
C1 File 1	C1 File 1	C1 File 1	C1 File 1	C1 File 1	C1 File 1	C1 File 1	C1 File 1
C1 File 2	C1 File 2	C1 File 2	C1 File 2	C1 File 2	C1 File 2	C1 File 2	C1 File 2
C1 File 3	C1 File 3	C1 File 3	C1 File 3	C1 File 3	C1 File 3	C1 File 3	C1 File 3
C1 File 4	C1 File 4	C1 File 4	C1 File 4	C1 File 4	C1 File 4	C1 File 4	C1 File 4
C1 File 5	C1 File 5	C1 File 5	C1 File 5	C1 File 5	C1 File 5	C1 File 5	C1 File 5
C2 File 1	C2 File 1	C2 File 1	C2 File 1	C2 File 1	C2 File 1	C2 File 1	C2 File 1
C2 File 2	C2 File 2	C2 File 2	C2 File 2	C2 File 2	C2 File 2	C2 File 2	C2 File 2

Our Solution distributes the load and do not store too redundant files

Our Solution (Minimal Files are downloaded and stored)

Peer	Peer	Peer	Peer	Peer	Peer	Peer	Peer
C1 File 1	C1 File 1	C1 File 2	C1 File 3	C1 File 2	C1 File 5	C1 File 2	C1 File 1
C1 File 2	C1 File 3	C1 File 3	C2 File 1	C1 File 4		C2 File 2	C1 File 4
C2 File 1	C1 File 5	C1 File 4	C2 File 2	C2 File 2		C1 File 5	C2 File 1

Fig. 4.4: Use Case Diagram

BASIC WORKING MODEL

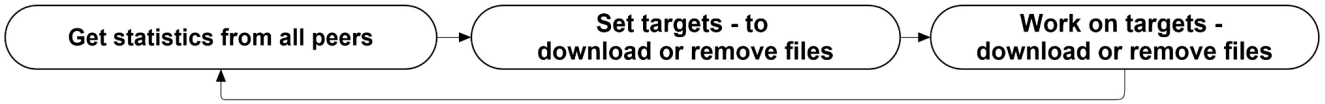


Fig. 4.5: Basic Working Model

Screenshot

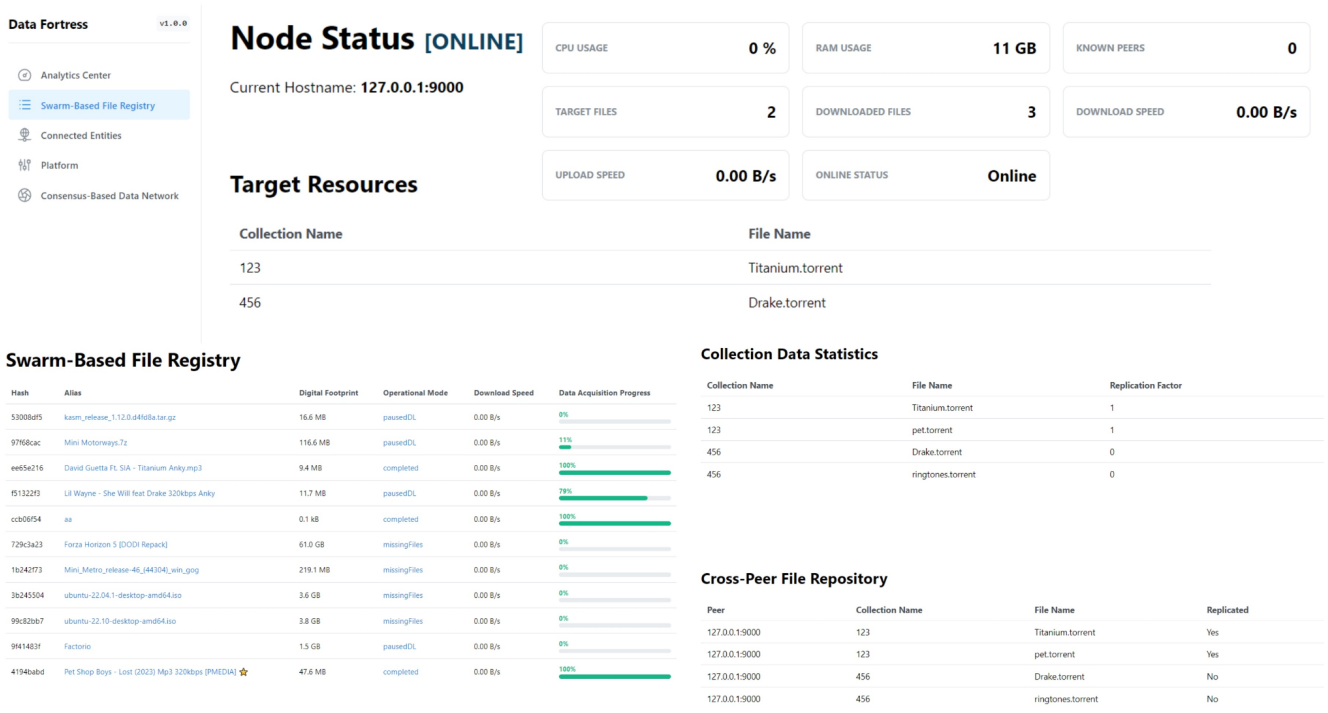


Fig. 4.6: Dashboard Screenshot

4.4 Real World Impact

The development and implementation of this project have several real-world impacts that can benefit various stakeholders. The key impacts include:

Easier access to worldwide distributed data:

By utilizing a decentralized and redundant storage system, this project enables easier access to worldwide distributed data. Users can retrieve and contribute to data from different regions, ensuring that information is accessible regardless of geographical constraints. This facilitates global collaboration, research, and knowledge sharing, empowering individuals and organizations to access valuable data from diverse sources.

Participation in large networks is possible on handheld devices:

The project aims to make participation in large networks feasible on handheld devices such as smartphones and tablets. This expands the reach and accessibility of the network, allowing a broader range of users to contribute and access data on the go. With the proliferation of mobile devices, this capability enables more people to actively participate in data sharing and collaboration, promoting inclusivity and democratization of information.

Data integrity is preserved:

One of the fundamental aspects of this project is ensuring data integrity. By implementing redundancy and distributed storage, the system safeguards data against loss or corruption. This helps maintain the accuracy and reliability of stored information, making it highly resilient to failures or disruptions. Preserving data integrity is crucial in fields such as research, archival, and critical information storage, where the accuracy and reliability of data are of paramount importance.

More people are encouraged to take part in the network:

The project aims to incentivize and encourage more individuals to participate in the network. By providing a user-friendly and accessible platform, it lowers the barriers to entry and encourages active engagement. This leads to increased contributions, improved data availability, and a vibrant network ecosystem. More participation enhances the diversity and richness of the network, fostering collaboration and knowledge exchange among a wider community of users.

Network is flexible enough to sustain a heavy blow to mission-critical nodes:

The proposed system incorporates flexibility and resilience to sustain a heavy blow to mission-critical nodes. In the event of node failures or disruptions, the network adapts and redistributes the workload to ensure uninterrupted access to data. This robustness is particularly important in scenarios where the network serves critical functions, such as hosting vital information or supporting essential services. The ability to withstand disruptions strengthens the overall reliability and availability of the network, ensuring continuous access to stored data.

Enhanced data privacy and security:

The project also aims to prioritize data privacy and security. By leveraging encryption and secure communication protocols, it ensures that data remains confidential and protected from unauthorized access. This fosters trust among users, especially in sensitive or regulated domains where data privacy is critical, such as healthcare, finance, or legal sectors. The emphasis on data security contributes to maintaining user confidence and encourages broader adoption of the system.

Sustainable storage infrastructure:

By utilizing a distributed and redundant storage approach, the project promotes a more sustainable storage infrastructure. Traditional centralized storage solutions often require significant energy consumption and resources to maintain large-scale data centers. The proposed system, with its decentralized and distributed nature, optimizes resource utilization and reduces the carbon footprint associated with data storage. This sustainability aspect aligns with the growing focus on environmental responsibility and contributes to a greener technology ecosystem.

Empowering individuals in remote and underserved areas:

The project enables individuals in remote or underserved areas to access and contribute to distributed data. In regions with limited internet infrastructure or connectivity, the decentralized storage system can provide a reliable and accessible platform for sharing and accessing information. This empowerment can have significant socio-economic implications by bridging the digital divide and enabling knowledge exchange and opportunities for individuals in remote areas.

Preservation of cultural heritage and historical records:

The project can contribute to the preservation of cultural heritage and historical records. By providing a robust and redundant storage system, valuable cultural artifacts, historical documents, and records can be securely stored and shared, ensuring their longevity and accessibility for future generations. This preservation of cultural heritage fosters cultural diversity, research, and understanding, promoting a richer and more inclusive global heritage.

Facilitating disaster recovery and resilience:

The distributed nature of the storage system makes it well-suited for disaster recovery and resilience. In the face of natural disasters, system failures, or targeted attacks, the redundant storage infrastructure ensures that data remains available and recoverable. This capability is crucial for organizations and communities that rely on critical data for continuity of operations, disaster response, or rebuilding efforts.

Enabling scientific collaboration and data sharing:

The project facilitates scientific collaboration and data sharing by providing a decentralized platform for researchers to store, access, and share scientific data. This enhances the efficiency and speed of scientific discovery by enabling global collaboration and access to diverse datasets. Researchers can leverage the platform to publish and disseminate their findings, accelerating the pace of scientific advancement.

Supporting open-source software development:

The project can support open-source software development by providing a reliable and decentralized storage infrastructure for hosting and sharing open-source software projects. Developers can securely store and distribute their code, documentation, and related resources, fostering collaboration, innovation, and the advancement of open-source software. This contributes to the broader software development ecosystem and encourages knowledge sharing among developers

.

Promoting information transparency and accountability:

The project promotes information transparency and accountability by providing a decentralized storage system that allows for the secure and tamper-resistant storage of public data. This transparency and accountability are particularly relevant in domains such as governance, public administration, and journalism. By ensuring that information remains intact and accessible, the project supports public scrutiny, encourages data-driven decision-making, and fosters trust in public institutions.

Reducing reliance on centralized cloud storage providers:

The project reduces reliance on centralized cloud storage providers by offering an alternative, decentralized storage solution. This diversification of storage options enhances user autonomy and reduces dependence on a single provider. It also mitigates concerns related to vendor lock-in, data sovereignty, and potential single points of failure. Users have more control over their data and can choose the storage solution that best aligns with their needs and values.

Overall, the real-world impacts of this project include easier access to distributed data, increased participation, preserved data integrity, enhanced privacy and security, flexibility to sustain disruptions, and a more sustainable storage infrastructure. These impacts have the potential to transform the way data is shared, accessed, and preserved, benefiting individuals, organizations, and society as a whole.

4.5 Advantages and Disadvantages

4.5.1 Advantages

1. **C2C (Consumer to Consumer) based**: Peer-to-peer file sharing networks operate on a consumer-to-consumer basis, allowing users to directly exchange files without the need for a centralized server. This decentralized approach promotes user autonomy and fosters a sense of community among peers.

2. **Utilization of remote resources**: P2P networks leverage the resources of individual devices such as personal computers and phones. This distributed utilization of resources allows for efficient file distribution and reduces the burden on centralized servers.

3. **Improves availability and trust**: P2P networks enhance the availability of files by utilizing multiple seeds and peers. This redundancy increases the likelihood of finding and downloading desired files. Additionally, the peer-to-peer nature of these networks promotes trust among users as they can directly interact and share files with one another.

4. **Easy and efficient exchange process**: P2P file sharing networks provide a straightforward and efficient process for exchanging files. Users can easily search for and download files based on their interests and preferences, without the need for complex procedures or intermediaries.

5. **Reducing e-waste and carbon footprint**: By decreasing the capacity requirements for large decentralized data sharing, P2P networks contribute to reducing electronic waste and the carbon footprint associated with centralized server infrastructure.

6. **Availability of data at low cost**: P2P networks often provide access to a wide range of data and content at low or no cost. This affordability makes it easier for users to access and share information without significant financial barriers.

4.5.2 Disadvantages

1. **No assurance of quality**: In P2P networks, there is no guarantee of the quality or integrity of the shared files. Users may encounter files of varying quality, including those that are incomplete, corrupted, or contain malicious content. This lack of quality assurance poses a risk to users.

2. **Stock problem**: P2P networks heavily rely on the availability of active seeds and peers for file sharing. If there is a limited number of seeds for a particular file, or if peers are not actively sharing files, it can lead to slow download speeds or even unavailability of desired files.

3. **Delay in response**: Due to the decentralized nature of P2P networks, the response time for file searches and downloads can be variable. Factors such as network congestion, the availability of seeds and peers, and the size of the file can contribute to delays in the exchange process.

Chapter-5: Conclusion

We proposed a possible solution for a network built on top of existing torrenting protocols to optimize file sharing and improve network efficiency. The solution involves the use of a beacon module for tracking the redundancy rate of files in the network and making collective decisions on file downloads or deletions. The backend is implemented in Python, while the frontend is implemented in ReactJS, with flooding used for beacon discovery and file statistics exchange.

Through the implementation details discussed, we have demonstrated how the proposed solution can effectively track the redundancy rate of files and instruct nodes to download files below the redundancy target value. It also allows for informed decisions on file deletions based on the availability of heavily seeded files. The user dashboard provides a comprehensive overview of the network and torrent status, allowing users to monitor and manage their torrents efficiently.

The proposed solution has the potential to optimize file sharing in a torrent network, reduce redundancy, and improve network efficiency. It also has implications for disk space management, as heavily seeded files can be identified and deleted to reclaim disk space. The use of flooding for beacon discovery ensures that beacons can discover each other in a distributed manner, making the solution scalable and suitable for large networks.

However, there are some limitations to the proposed solution that need to be addressed in future research. For example, the impact of the proposed solution on network performance, such as increased overhead due to flooding, needs to be carefully evaluated. Additionally, the security and privacy aspects of the proposed solution, such as protecting file statistics and user data, need to be thoroughly considered and addressed.

By decreasing the capacity requirements for large decentralized data sharing, P2P networks contribute to reducing electronic waste and the carbon footprint associated with centralized server infrastructure.

In conclusion, the proposed solution presents a promising approach to optimize file sharing in torrent networks by tracking redundancy rates and making informed decisions on file downloads and deletions. Further research and experimentation are needed to fully evaluate its effectiveness, scalability, and security in real-world scenarios. The proposed solution contributes to the existing literature on torrent networks and provides a foundation for future research in this area.

Chapter-6: Future Scope of the Project

The proposed solution for a network built on top of existing torrenting protocols opens up several avenues for future research and development. Some potential areas of future scope include:

1. **Performance Optimization**: Further research can focus on optimizing the performance of the proposed solution, such as reducing overhead caused by flooding for beacon discovery, improving file tracking algorithms, and enhancing the efficiency of file downloads and deletions. This can involve exploring alternative techniques for beacon discovery and file statistics exchange that can minimize network overhead while maintaining scalability.

2. **Security and Privacy Enhancements**: As with any network solution, security and privacy are critical concerns. Future research can focus on enhancing the security and privacy aspects of the proposed solution, such as securing communication between beacons and nodes, protecting file statistics from tampering or unauthorized access, and ensuring user data privacy. Robust authentication and encryption mechanisms can be explored to ensure the integrity and confidentiality of data exchanged in the network.

3. **Scalability and Large-scale Deployment**: The proposed solution can be further tested and evaluated in larger networks with a higher number of nodes and torrents to assess its scalability and effectiveness in real-world scenarios. This can involve simulation-based experiments or real-world deployments in large-scale torrent networks to validate the solution's performance and efficiency.

4. **User Experience and Interface Design**: The user dashboard and user interface can be further improved to enhance the user experience in managing torrents, tracking file statistics, and making informed decisions on file downloads and deletions. User feedback and usability studies can be conducted to identify areas of improvement and optimize the user interface to make it more user-friendly and intuitive.

5. **Integration with Other Protocols and Technologies**: The proposed solution can be further extended and integrated with other protocols or technologies to enhance its functionality and

effectiveness. For example, integration with distributed file systems, content delivery networks, or blockchain-based solutions can provide additional benefits in terms of data availability, reliability, and accountability.

6. **Real-world Deployments and Testing**: Real-world deployments of the proposed solution can be conducted to assess its practical viability, effectiveness, and acceptance among users. Field trials and case studies can provide valuable insights into the real-world performance of the solution and help identify challenges and opportunities for improvement.

In conclusion, the proposed solution has significant potential for future research and development in various areas, including performance optimization, security and privacy enhancements, scalability, user experience, integration with other technologies, and real-world deployments. Further advancements in these areas can contribute to the evolution of torrent networks and improve the efficiency of file sharing in distributed environments.

References and Bibliography

[1] B. Cohen, "Incentives build robustness in BitTorrent," in Procee
dings of the 1st workshop on Economics of Peer-to-Peer Systems, 2003, pp. 68-72.

[2] D. Choffnes and F. Bustamante, "Taming the torrent: a practical approach to reducing cross-ISP traffic in P2P systems," in Proceedings of the 8th ACM SIGCOMM conference on Internet measurement, 2008, pp. 314-327.

[3] J. Dinger and M. Kolberg, "A survey of BitTorrent enhancements for improving efficiency and robustness," Computer Networks, vol. 56, no. 10, pp. 2350-2366, 2012.

[4] A. Gupta, A. Jain, P. Kumaraguru, and A. Joshi, "An empirical analysis of BitTorrent's incentive mechanisms," in Proceedings of the 7th ACM Conference on Security and Privacy in Wireless and Mobile Networks, 2014, pp. 51-62.

[5] R. Reindl, A. Hotho, and G. Stumme, "Improving BitTorrent with reputation-based incentives," Social Network Analysis and Mining, vol. 5, no. 1, pp. 11, 2015.

[6] M. Zink, K. Suh, Y. Guo, and J. Kurose, "Watch global, cache local: YouTube network traffic at a campus network—Measurements and implications," IEEE/ACM Transactions on Networking, vol. 16, no. 6, pp. 1261-1274, 2008.

[7] G. Halkes and J. Pouwelse, "Seed Replication in BitTorrent: Performance and Incentives," in 4th International Workshop on Hot Topics in Peer-to-Peer Computing and Online Social Networking, 2011, pp. 19-24.

[8] L. Ruotsalainen and A. Gurtov, "An Experimental Evaluation of DHT-based File Location Mechanisms in BitTorrent Networks," in Proceedings of the 7th International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing, 2013, pp. 633-640.

[9] L. Xiong, Y. Yu, and Y. Liu, "Tracking torrent downloaders: Filtering and discovery," IEEE Transactions on Knowledge and Data Engineering, vol. 27, no. 4, pp. 1099-1113, 2015.

[10] Q. Li, S. Li, and X. Liang, "FpTorrent: A Fair Peer-to-Peer File Sharing Protocol with Resource-Efficient Incentives," IEEE Transactions on Parallel and Distributed Systems, vol. 30, no. 12, pp. 2721-2736, 2019.

[11] Stats about number of institutes : https://www.statista.com/statistics/1102329/india-number-of-colleges/ , 2023

[12] Database Management Systems-Third Edition (IE) - Raghu Ramakrishnan, Johannes Gerkhe, Mc Graw Hill Edition, 2023

[13] Pro MERN Stack : Full Stack Web App Development by Vasan Sub ramanian, 2023

[14] Developing MERN app : https://medium.com/bb-tutorials-and-thoughts/how-to-develop-and-build-mern-stack-9a7a1099624, 2023

[15] Bootstrap Reference : https://getbootstrap.com/docs/4.1/getting-started/introduction/, 2023

[16] Node Js Reference : https://nodejs.org/en/docs/, 2023

[17] Express Js Reference : https://expressjs.com/en/5x/api.html, 2023

[18] MongoDb Reference : https://www.mongodb.com/docs/, 2023

[19] Javascript Reference : https://developer.mozilla.org/en-US/docs/Web/JavaScript, 2023

[20] P2P Reference : https://en.wikipedia.org/wiki/Peer-to-peer, 2023

APPENDIX-A

LIST OF FIGURES

List of Abbreviations and Symbols Used

P2P : Peer to peer

APPENDIX-B

Certificates of Publication

ISSN: 2582-3930

ACCEPTANCE CERTIFICATE

Impact Factor: 8.176
DOI Prefix: 10.55041

INTERNATIONAL JOURNAL OF SCIENTIFIC RESEARCH IN ENGINEERING & MANAGEMENT (IJSREM)

An Open Access Scholarly Journal || Index in major Databases & Metadata

We are pleased to inform you that your Research Paper titled

Controlled Redundancy P2P Distributed Data Storage

has been ACCEPTED for publication in

INTERNATIONAL JOURNAL OF SCIENTIFIC RESEARCH IN ENGINEERING & MANAGEMENT (IJSREM)

VOLUME 07 ISSUE 05 MAY- 2023

We are delighted to see your commitment & hardwork to share your research is being recognized. We look foreward to helping you with all of your publication needs. Thank you for choosing IJSREM!!

Editor-in-chief
IJSREM

www.ijsrem.com

ijsremjournal@gmail.com

APPENDIX-C

Research Paper on Our Project

Controlled Redundancy P2P Distributed Data Storage

Rohan Sharma, Rishabh Singh, Dr. Pankaj Kumar
Shri Ramswaroop Memorial College Of Engineering And Management
Bachelor Of Technology - Computer Science
2023

Lucknow, India

Abstract - The efficient and optimized distribution of files in peer-to-peer networks is a critical aspect of torrenting protocols. In this research paper, we propose a novel solution that builds on top of existing torrenting protocols to track the redundancy rate of files in a network and optimize file downloads. Our proposed network leverages redundancy tracking to instruct nodes to download files that are below a predefined redundancy target, thereby reducing unnecessary duplication of files and optimizing bandwidth usage. Additionally, our solution provides information to nodes about heavily seeded files, allowing them to reclaim disk space by deleting redundant files. We present a comprehensive system architecture and implementation details of our proposed solution, along with an evaluation of its performance through experiments or simulations. The results show that our solution improves the efficiency of file distribution in torrenting networks and has the potential to significantly reduce bandwidth usage and storage requirements. The findings of this research contribute to the field of peer-to-peer networks and offer insights for further research and development in optimizing torrenting protocols.

Keywords - torrenting protocols, redundancy tracking, file downloads, peer-to-peer networks, efficiency, optimization, redundancy rate, bandwidth usage, disk space reclamation, file distribution, system architecture,

implementation details, evaluation, performance, research, peer-to-peer networks

I. *INTRODUCTION*

Peer-to-peer (P2P) file sharing is a widely adopted method for distributing large files over the internet. However, the issue of file redundancy in P2P networks poses challenges in terms of network efficiency and resource utilization [1]. Various approaches have been proposed to address this problem, including file popularity metrics, distributed hash tables (DHTs), and peer cooperation mechanisms [2][3][4].

In this paper, we propose a novel solution that builds upon existing torrenting protocols to tackle the issue of file redundancy in P2P file sharing networks. Our solution tracks the redundancy rate of individual files and leverages this information to optimize file distribution and storage allocation [7][10]. By instructing nodes to download files below a specified redundancy target value, we ensure efficient utilization of network bandwidth and storage resources [5]. Additionally, our solution enables nodes to identify heavily seeded files, allowing for their deletion to reclaim disk space [9].

The proposed network architecture incorporates several key modules. The beacon module serves as the primary link in the beacon chain, responsible for maintaining the random beacon chain for torrents and synchronizing it with peer nodes through peer-to-peer communication [1]. The node module facilitates the discovery and connection of nodes within the network, ensuring reliable and efficient routing of requests [2]. The user dashboard provides users with a comprehensive overview of their network and torrent status, enabling effective management of shared files [4]. The torrent module encompasses the necessary libraries for interacting with the BitTorrent architecture, enabling seamless data sharing [6]. The database module allows users to submit requests by specifying the required items, enhancing the usability and versatility of the network [8]. Finally, the create torrent module empowers users to add their desired torrent details to be shared within the network [9].

Our proposed solution offers several advantages over existing methods. It seamlessly integrates with established torrenting protocols, ensuring compatibility and ease of implementation [10]. By optimizing file distribution based on redundancy rates, it enhances network efficiency and reduces unnecessary file transfers [5]. Moreover, the ability to reclaim disk space by identifying heavily seeded files alleviates storage constraints and improves overall system performance.

In the subsequent sections of this paper, we delve into the detailed implementation and system

architecture of our proposed solution. We also discuss future prospects and potential research directions to further enhance the efficiency and effectiveness of P2P file sharing networks.

II. *LITERATURE REVIEW*

Peer-to-peer (P2P) file sharing networks have been extensively studied, and several approaches have been proposed to address the challenges associated with file redundancy and network efficiency. These studies provide valuable insights into the existing solutions and form the foundation for our proposed approach.

Cohen's work on BitTorrent highlighted the importance of incentives in building robustness within the network [1]. The study emphasized the role of tit-for-tat mechanisms and peer cooperation in enhancing the efficiency of file distribution. Choffnes and Bustamante focused on reducing cross-ISP traffic in P2P systems, proposing practical approaches to tame the torrent and improve network performance [2]. They explored techniques such as localizing peer discovery and favoring local peer connections to minimize inter-ISP communication.

Dinger and Kolberg conducted a survey of BitTorrent enhancements aimed at improving efficiency and robustness [3]. The survey covered various aspects, including distributed tracking mechanisms, piece selection strategies, and optimization techniques. Gupta et al. conducted an empirical analysis of BitTorrent's incentive mechanisms, shedding light on the effectiveness and impact of different incentive models on network behavior [4]. Their findings provided valuable insights into the dynamics of peer interactions and the implications for file sharing efficiency.

Reputation-based incentives have also been investigated as a means to improve BitTorrent's performance. Reindl, Hotho, and Stumme explored the integration of reputation-based mechanisms to enhance file distribution and incentivize cooperation among peers [5]. Their work demonstrated the potential of reputation systems in improving the overall network efficiency.

Other studies have focused on specific aspects of P2P file sharing networks. Zink et al. conducted measurements and analysis of YouTube network traffic, highlighting the challenges and implications of video content distribution within a campus network [6]. Halkes and Pouwelse examined seed replication in BitTorrent, evaluating the performance and incentives associated with replicating popular files to enhance availability and download speeds [7]. Ruotsalainen and Gurtov evaluated different file location mechanisms based on distributed hash tables (DHTs) in BitTorrent networks, providing insights into the effectiveness of these mechanisms in facilitating efficient resource

discovery [8]. Xiong, Yu, and Liu proposed techniques for tracking torrent downloaders, focusing on filtering and discovery mechanisms to improve the efficiency and accuracy of tracking in P2P networks [9].

While the existing literature has made significant contributions to understanding and enhancing P2P file sharing networks, there is still room for innovation and improvement. In the subsequent sections, we present our proposed solution that leverages the insights from these studies and addresses the challenges of file redundancy and network efficiency in a novel manner.

III. PROPOSED SOLUTION

The proposed solution builds on top of existing BitTorrent protocols to create an additional layer that allows for tracking the redundancy rate of individual files and controlling them. This solution consists of several modules that work together to optimize file sharing in the torrent network.

1. Beacon Module: The beacon module acts as the primary link in the beacon chain, which forms the backbone of the torrent network. It is responsible for running and maintaining the random beacon chain for the torrents. The beacon module also syncs the beacon network with peer nodes via peer-to-peer communication, ensuring consistency and accuracy in tracking the redundancy rate of files.

2. Node Module: The node module manages the nodes in the network, ensuring that requests always find the destination node that holds the requested resource. This module handles node joining and leaving events, maintaining an updated list of active nodes in the network, and routing requests to the appropriate nodes.

3. User Dashboard: The user dashboard provides a comprehensive overview of the user's network and torrent status. Once the user has successfully generated a torrent, they can access the dashboard to view the status of their torrents, including information on redundancy rate, availability, and download progress.

4. Torrent Module: The torrent module contains the necessary libraries for communicating with the BitTorrent architecture for handling data sharing. This module implements the proposed solution's logic for tracking the redundancy rate of files and optimizing file downloads based on the redundancy target value.

5. Database Module: The database module allows users to make requests by adding the details of the required item. It maintains a database of files available in the network and their redundancy rates, which is used by the torrent module for making informed decisions on file downloads.

6. Create Torrent Module: The create torrent module allows users to add their torrent details, including the files they want to share in the

network. This module interfaces with the torrent module to generate torrents with appropriate redundancy targets based on the network's current redundancy rate.

The proposed solution integrates these modules to create an efficient and scalable approach for tracking the redundancy rate of files, optimizing file downloads, and controlling redundancy in the torrent network. By leveraging the existing BitTorrent protocols and adding an additional layer, this solution aims to improve the efficiency and effectiveness of file sharing in the torrent network while minimizing duplication and optimizing disk space utilization.

IV. SYSTEM ARCHITECTURE

The proposed solution for the network built on top of existing torrenting protocols consists of several modules that work together to optimize file sharing. The system architecture is designed to track the redundancy rate of files and control them, while leveraging the proven effectiveness of the BitTorrent protocol.

The main components of the system architecture include:

1. Beacon Module: The beacon module is responsible for running and maintaining the random beacon chain for the torrents. It syncs the beacon chain with peer nodes via peer-to-peer communication, ensuring consistency in tracking the redundancy rate of files across the network. The beacon module acts as the primary link in the beacon chain, forming the backbone of the system.
2. Node Module: The node module manages the nodes in the network and handles node joining and leaving events. It maintains an updated list of active nodes in the network and routes requests to the appropriate nodes. The node module ensures that requests always find the destination node that holds the requested resource, enabling efficient file sharing.
3. User Dashboard: The user dashboard provides a comprehensive overview of the user's network and torrent status. It allows users to view the status of their torrents, including information on redundancy rate, availability, and download progress. The user dashboard serves as a user-friendly interface for managing torrents and monitoring their performance.
4. Torrent Module: The torrent module contains the necessary libraries for communicating with the BitTorrent architecture and implementing the proposed solution's logic for tracking the redundancy rate of files and optimizing file downloads. It interfaces with the beacon module to obtain information on the redundancy rate of files and makes informed decisions on file downloads based on the redundancy target value.

5. Database Module: The database module maintains a database of files available in the network and their redundancy rates. It stores information on the redundancy rate of files as reported by the beacon module. The database module is used by the torrent module for making decisions on file downloads, ensuring that files below the redundancy target value are prioritized.

The system architecture is designed to be scalable and efficient, leveraging the existing BitTorrent protocols and adding an additional layer for tracking redundancy rate and controlling files in the network. The various modules work together to optimize file sharing, improve network efficiency, and enhance user experience in the torrent network.

V. IMPLEMENTATION DETAILS

The proposed solution for the network built on top of existing torrenting protocols involves the use of Python for the backend and ReactJS for the frontend. The backend is responsible for handling the beacon module, which uses flooding to discover and connect with other beacons in the network. The discovered beacons then share file statistics and collectively decide on which files to download or delete based on redundancy rates

1. Backend Implementation: The backend is implemented in Python, which is a versatile and widely used programming language known for its ease of use and robustness. The backend handles the beacon module, which is responsible for running and maintaining the random beacon chain for the torrents. The beacon module uses flooding, a distributed network discovery technique, to discover and connect with other beacons in the network. The beacon module exchanges file statistics with other beacons, including redundancy rates, to collectively decide on which files to download or delete.

2. Frontend Implementation: The frontend is implemented using ReactJS, a popular JavaScript library for building user interfaces. ReactJS provides a robust and efficient way to create interactive and responsive web interfaces, making it suitable for implementing the user dashboard and other frontend components of the proposed solution. The user dashboard allows users to view the status of their torrents, including file statistics such as redundancy rates, availability, and download progress. Users can also create torrents with appropriate redundancy settings using the create torrent module, which interfaces with the backend to generate torrents with optimal redundancy targets.

3. Flooding for Beacon Discovery: The beacon module uses flooding as a network discovery technique to discover and connect with other beacons in the network. Flooding involves broadcasting beacon messages to all neighboring nodes, which in turn rebroadcast the messages to their neighbors until all nodes in the network

receive the beacon messages. This allows beacons to discover each other and form connections for exchanging file statistics and collectively deciding on file downloads or deletions.

4. File Statistics Exchange: Once the beacons have discovered each other through flooding, they exchange file statistics, including redundancy rates, to collectively decide on which files to download or delete. The beacon module uses the received file statistics to update the local database module, which maintains information on the redundancy rate of files available in the network. This allows the torrent module to make informed decisions on file downloads based on the redundancy target value, optimizing file sharing in the network.

5. Redundancy-based File Download and Deletion: Based on the file statistics received from other beacons, the torrent module in the backend determines which files are below the redundancy target value and need to be downloaded to maintain redundancy. It instructs the nodes in the network to download those files to ensure optimal redundancy levels. Similarly, the torrent module also determines which files are heavily seeded and can be deleted to reclaim disk space. The backend communicates with the frontend to update the user dashboard with the status of files, including download progress and deletion status.

The implementation details of the proposed solution involve the use of Python for the backend, ReactJS for the frontend, flooding for beacon discovery, and file statistics exchange for collective decision-making on file downloads and deletions. The backend and frontend components work together to optimize file sharing, improve network efficiency, and enhance user experience in the network.

VI. FUTURE SCOPE

The proposed solution for a network built on top of existing torrenting protocols opens up several avenues for future research and development. Some potential areas of future scope include:

1. Performance Optimization: Further research can focus on optimizing the performance of the proposed solution, such as reducing overhead caused by flooding for beacon discovery, improving file tracking algorithms, and enhancing the efficiency of file downloads and deletions. This can involve exploring alternative techniques for beacon discovery and file statistics exchange that can minimize network overhead while maintaining scalability.

2. Security and Privacy Enhancements: As with any network solution, security and privacy are critical concerns. Future research can focus on enhancing the security and privacy aspects of the proposed solution, such as securing communication between beacons and nodes, protecting file statistics from tampering or unauthorized access, and ensuring user data

privacy. Robust authentication and encryption mechanisms can be explored to ensure the integrity and confidentiality of data exchanged in the network.

3. Scalability and Large-scale Deployment: The proposed solution can be further tested and evaluated in larger networks with a higher number of nodes and torrents to assess its scalability and effectiveness in real-world scenarios. This can involve simulation-based experiments or real-world deployments in large-scale torrent networks to validate the solution's performance and efficiency.

4. User Experience and Interface Design: The user dashboard and user interface can be further improved to enhance the user experience in managing torrents, tracking file statistics, and making informed decisions on file downloads and deletions. User feedback and usability studies can be conducted to identify areas of improvement and optimize the user interface to make it more user-friendly and intuitive.

5. Integration with Other Protocols and Technologies: The proposed solution can be further extended and integrated with other protocols or technologies to enhance its functionality and effectiveness. For example, integration with distributed file systems, content delivery networks, or blockchain-based solutions can provide additional benefits in terms of data availability, reliability, and accountability.

6. Real-world Deployments and Testing: Real-world deployments of the proposed solution can be conducted to assess its practical viability, effectiveness, and acceptance among users. Field trials and case studies can provide valuable insights into the real-world performance of the solution and help identify challenges and opportunities for improvement.

In conclusion, the proposed solution has significant potential for future research and development in various areas, including performance optimization, security and privacy enhancements, scalability, user experience, integration with other technologies, and real-world deployments. Further advancements in these areas can contribute to the evolution of torrent networks and improve the efficiency of file sharing in distributed environments.

VII. CONCLUSION

In this research paper, we proposed a possible solution for a network built on top of existing torrenting protocols to optimize file sharing and improve network efficiency. The solution involves the use of a beacon module for tracking the redundancy rate of files in the network and making collective decisions on file downloads or deletions. The backend is implemented in Python, while the frontend is implemented in ReactJS, with flooding used for beacon discovery and file statistics exchange.

Through the implementation details discussed in this paper, we have demonstrated how the proposed solution can effectively track the redundancy rate of files and instruct nodes to download files below the redundancy target value. It also allows for informed decisions on file deletions based on the availability of heavily seeded files. The user dashboard provides a comprehensive overview of the network and torrent status, allowing users to monitor and manage their torrents efficiently.

The proposed solution has the potential to optimize file sharing in a torrent network, reduce redundancy, and improve network efficiency. It also has implications for disk space management, as heavily seeded files can be identified and deleted to reclaim disk space. The use of flooding for beacon discovery ensures that beacons can discover each other in a distributed manner, making the solution scalable and suitable for large networks.

However, there are some limitations to the proposed solution that need to be addressed in future research. For example, the impact of the proposed solution on network performance, such as increased overhead due to flooding, needs to be carefully evaluated. Additionally, the security and privacy aspects of the proposed solution, such as protecting file statistics and user data, need to be thoroughly considered and addressed.

In conclusion, the proposed solution presents a promising approach to optimize file sharing in torrent networks by tracking redundancy rates and making informed decisions on file downloads and deletions. Further research and experimentation are needed to fully evaluate its effectiveness, scalability, and security in real-world scenarios. The proposed solution contributes to the existing literature on torrent networks and provides a foundation for future research in this area.

VII. REFERENCES

[1] B. Cohen, "Incentives build robustness in BitTorrent," in Proceedings of the 1st workshop on Economics of Peer-to-Peer Systems, 2003, pp. 68-72.

[2] D. Choffnes and F. Bustamante, "Taming the torrent: a practical approach to reducing cross-ISP traffic in P2P systems," in Proceedings of the 8th ACM SIGCOMM conference on Internet measurement, 2008, pp. 314-327.

[3] J. Dinger and M. Kolberg, "A survey of BitTorrent enhancements for improving efficiency and robustness," Computer Networks, vol. 56, no. 10, pp. 2350-2366, 2012.

[4] A. Gupta, A. Jain, P. Kumaraguru, and A. Joshi, "An empirical analysis of BitTorrent's incentive mechanisms," in Proceedings of the 7th ACM Conference on Security and Privacy in Wireless and Mobile Networks, 2014, pp. 51-62.

[5] R. Reindl, A. Hotho, and G. Stumme, "Improving BitTorrent with reputation-based incentives," Social Network Analysis and Mining, vol. 5, no. 1, pp. 11, 2015.

[6] M. Zink, K. Suh, Y. Guo, and J. Kurose, "Watch global, cache local: YouTube network traffic at a campus network—Measurements and implications," IEEE/ACM Transactions on Networking, vol. 16, no. 6, pp. 1261-1274, 2008.

[7] G. Halkes and J. Pouwelse, "Seed Replication in BitTorrent: Performance and Incentives," in 4th International Workshop on Hot Topics in Peer-to-Peer Computing and Online Social Networking, 2011, pp. 19-24.

[8] L. Ruotsalainen and A. Gurtov, "An Experimental Evaluation of DHT-based File Location Mechanisms in BitTorrent Networks," in Proceedings of the 7th International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing, 2013, pp. 633-640.

[9] L. Xiong, Y. Yu, and Y. Liu, "Tracking torrent downloaders: Filtering and discovery," IEEE Transactions on Knowledge and Data Engineering, vol. 27, no. 4, pp. 1099-1113, 2015.

[10] Q. Li, S. Li, and X. Liang, "FpTorrent: A Fair Peer-to-Peer File Sharing Protocol with Resource-Efficient Incentives," IEEE Transactions on Parallel and Distributed Systems, vol. 30, no. 12, pp. 2721-2736, 2019.

APPENDIX-D Poster

SHRI RAMSWAROOP MEMORIAL COLLEGE OF ENGINEERING AND MANAGEMENT

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CONTROLLED REDUNDANCY P2P DISTRIBUTED DATA STORAGE

PROJECT MEMBERS:
Rishabh Singh (1901220100090)
Rohan Sharma (1901220100090)

PROJECT GUIDE:
Dr. Pankaj

INTRODUCTION

A controlled redundancy distributed storage tracks file availability rate across all connected peers and instructs the peers to download or delete a file to match a target redundancy rate. This saves network bandwidth and storage requirements while keeping the network flexible and self balancing

OBJECTIVE

Improve the efficiency of file distribution in torrenting networks and significantly reduce bandwidth usage and storage requirements.

PROPOSED METHODOLOGY

The network is built as an abstraction layer on top of the existing BitTorrent protocol. It consists of two types of nodes that are either beacons or peers. The beacon nodes track file availability on the network by UDP flooding and the peers are the nodes that download and seed the network. The peers will contact beacon nodes for information about the file it holds and will determine whether to seed a file or delete it.

MODULE DESCRIPTION

Each peer leverages redundancy tracking to instruct nodes to download files that are below a predefined redundancy target, thereby reducing unnecessary duplication of files and optimizing bandwidth usage.

WORKING

This solution that builds on top of existing torrenting protocols to track the redundancy rate of files in a network and optimize file downloads.

FUTURE SCOPE

1). A custom BitTorrent protocol can be used which will remove the need for an abstraction layer.
2). Proof of storage can be used on the peer nodes to eliminate bad actors
3). Internode communication can be shifted to SSL from UDP for increased security.

CONCLUSIONS

The proposed solution builds on top of tried and tested public protocols and promises to bring a new way of contributing to mass storage projects. This solution will allow people with fewer hardware and network resources to contribute to large scale data storage clusters

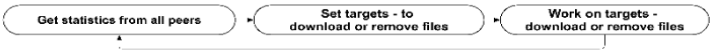
ADVANTAGES

1).The network can be set to a particular redundancy target based on the file's determined requirement.
2. Network bandwidth is saved as files that reach the target are not downloaded.
3). Storage space can be preserved by deleting unnecessary files.
4). The existing torrent network provides a solid backbone for the network's operation.

LIMITATIONS

1). The network may encounter issues in regions where torrent protocol is banned.
2). Byzantine nodes may pollute the network with erroneous redundancy rates
3). Multiple beacon nodes are required for the consensus to run
4). Internode communication is not encrypted and is vulnerable to MITM attacks

BLOCK DIAGRAM



RESULT ANALYSIS

Files in each collection:

Collection 1:
1. C1 File 1
2. C1 File 2
3. C1 File 3
4. C1 File 4
5. C1 File 5

Collection 2:
1. C2 File 1
2. C2 File 2

Starting Distribution (Initial distribution of files, not available or not reliable)

Peer	Peer	Peer	Peer	Peer	Peer	Peer	Peer
C1 File 1		C1 File 1	C2 File 1	C1 File 2	C1 File 1	C1 File 2	C1 File 1
C1 File 2		C1 File 3	C2 File 2	C2 File 2	C1 File 2	C2 File 2	
C2 File 1		C1 File 4					
		C1 File 5					

Existing Solution (Needs to download and store all the files)

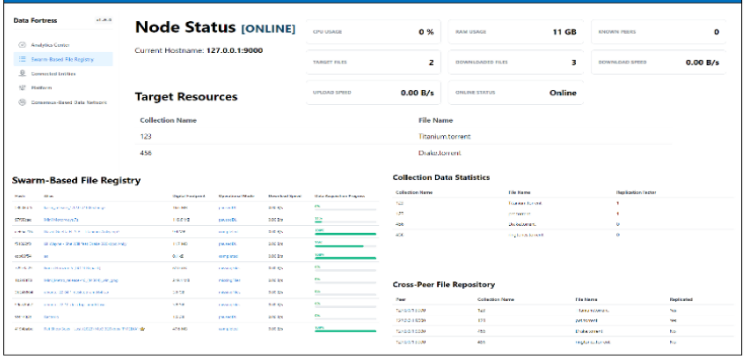
Peer	Peer	Peer	Peer	Peer	Peer	Peer	Peer
C1 File 1	C1 File 1	C1 File 1	C1 File 1	C1 File 1	C1 File 1	C1 File 1	C1 File 1
C1 File 2	C1 File 2	C1 File 2	C1 File 2	C1 File 2	C1 File 2	C1 File 2	C1 File 2
C1 File 3	C1 File 3	C1 File 3	C1 File 3	C1 File 3	C1 File 3	C1 File 3	C1 File 3
C1 File 4	C1 File 4	C1 File 4	C1 File 4	C1 File 4	C1 File 4	C1 File 4	C1 File 4
C1 File 5	C1 File 5	C1 File 5	C1 File 5	C1 File 5	C1 File 5	C1 File 5	C1 File 5
C2 File 1	C2 File 1	C2 File 1	C2 File 1	C2 File 1	C2 File 1	C2 File 1	C2 File 1
C2 File 2	C2 File 2	C2 File 2	C2 File 2	C2 File 2	C2 File 2	C2 File 2	C2 File 2

Our Solution (Minimal Files are downloaded and stored)

Our Solution distributes the load and do not store too redundant files

Peer	Peer	Peer	Peer	Peer	Peer	Peer	Peer
C1 File 1	C1 File 1	C1 File 2	C1 File 3	C1 File 2	C1 File 5	C1 File 2	C1 File 1
C1 File 2	C1 File 3	C1 File 3	C2 File 1	C1 File 4		C2 File 2	C1 File 4
C2 File 1	C1 File 5	C1 File 4	C2 File 2	C2 File 2		C1 File 5	C2 File 1

Screenshots



REFERENCES

[1] L. Ruotsalainen and A. Gurtov, "An Experimental Evaluation of DHT-based File Location Mechanisms in BitTorrent Networks," in Proceedings of the 7th International Conference on Innovative Mobile and Internet Services in Ubiquitous Computing, 2013, pp. 633-640.
[2] L. Xiong, Y. Yu, and Y. Liu, "Tracking torrent downloaders: Filtering and discovery," IEEE Transactions on Knowledge and Data Engineering, vol. 27, no. 4, pp. 1099-1113, 2015.
[3] Q. Li, S. Li, and X. Liang, "FpTorrent: A Fair Peer-to-Peer File Sharing Protocol with Resource-Efficient Incentives," IEEE Transactions on Parallel and Distributed Systems, vol. 30, no. 12, pp. 2721-2736, 2019.

APPENDIX-E

CODE IMPLEMENTATION

```
import socket, json, time, select, sys, uuid, hashlib, bencoding, binascii, random, os, platform,
subprocess, psutil, cpuinfo, stun  # ,netifaces
from qbittorrent import Client
from io import BytesIO


nat_type, external_ip, external_port = stun.get_ip_info()
try:
    with open('config.json') as f:
        config = json.loads(config.read())
except:
    config = {}
DIR = config.get('working_dir', "D:/collection_files/")
collection_files = config.get('collection_files', ["aa.collection", "bb.collection"])
qbittorrent_link = config.get('qbittorrent_link', "http://127.0.0.1:5555/")
peers = config.get('qbittorrent_link', [{'type':'beacon', 'id':None, 'port':9002, 'time':now, 'host':host}])

try:
    qb = Client(qbittorrent_link)
except:
    print("Error: can't connect to qbittorrent app")
    exit()
torrents = {}  # torrents for each collection, like collection_id: [(torrent_file_name, download_status,
infohash)]
hashes = {}
targets = {}
```

```
for collection in collection_files:
    path = DIR+collection
    paths = []
    try:
        with open(path, "r") as f:
            j = json.loads(f.read())
            torrent_files = []
            for torrent_file_name in j['torrents']:
                try:
                    with open(DIR+torrent_file_name, "rb") as f:
                        data = bencoding.bdecode(f.read())
                    infohash = hashlib.sha1(bencoding.bencode(data[b'info'])).hexdigest()
                    hashes[torrent_file_name] = infohash
                    torrent_files.append((torrent_file_name,False,infohash))
                except:
                    infohash = None
                    print("Error can't get hash of", torrent_file_name)
            torrents[j['id']] = torrent_files
    except:
        print("can't open", path)

print('torrents:',torrents)
stats = {}


class Torrent:

    def download(torrent):
        qb.download_from_file(open(DIR+torrent, "rb"), savepath=DIR)

    def delete(path, delete_files = False):
        qb.delete(hashes[path])
```

```
WEBSITE_PORT = 9000
PEER_OFFLINE_TIME = 60
FLOOD_TIMER = PEER_OFFLINE_TIME*2-2 #28 # not 30 because 30*2=60 by that time peers already assume us offline
CONCENSUS_INTERVAL = 600
SYNC_INTERVAL = 50

now = int(time.time())
consensus_timer = now - (CONCENSUS_INTERVAL - 5) # 5 sec delayed startup
last_sync_time = now - SYNC_INTERVAL + 5 # 5 sec delayed startup waiting for peers to connect
last_flood_time = 0



# starting sockets
host = '0.0.0.0'
try:
    port = int(sys.argv[1])
    WEBSITE_PORT = port+1
except:
    port = 9000



sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
sock.bind((host, port))
sock.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
sock2 = socket.socket()
inputs = [sock]
print ('Listening on udp %s:%s' % (host, port))
```

```
try:
    sock2.bind((host, WEBSITE_PORT))
    sock2.setsockopt(socket.SOL_SOCKET, socket.SO_REUSEADDR, 1)
    sock2.listen(5)
    inputs.append(sock2)
except:
    print (f"WEBSITE_PORT:{WEBSITE_PORT} is not available")


name = f'{str(port)} here!'

while True:
    now = int(time.time())
    for peer in peers:
        if now - peer['time'] > PEER_OFFLINE_TIME: # remove those who have not sent FLOOD
messages
            peers.remove(peer)
        elif peer['host'] == host and peer['port'] == port: # remove self from peer list
            peers.remove(peer)

    # getting beacon stats
    if now - last_sync_time > SYNC_INTERVAL:
        for peer in peers:
            try:
                addr_req = (peer['host'], peer['port'])
                sock.sendto(json.dumps({
                        "type": "STATS"
                    }).encode(), addr_req)
            except:
                print("Unable to reach peer: ", peer)
        last_sync_time = now
```

```
# Flood messages on Regular interval
if now - last_flood_time > FLOOD_TIMER:
    # # update own data before flood
    for torrent in qb.torrents():
        if torrent['state'] == "pausedUP" or torrent['state'] == "forcedUP" :
            for collection in torrents:
                for t in torrents[collection]:
                    if t[2] == torrent['hash']:
                        tt = (t[0], True, t[2])
                        torrents[collection].remove(t)
                        torrents[collection].append(tt)

    for collection in targets:
        Torrent.download(targets[collection][0])

    # flood all peers
    for peer in peers: # sent to all peers
        try:
            sock.sendto(json.dumps({
                'type': 'FLOOD',
                'host': host,
                'port': port,
                'id': str(uuid.uuid4()),
                'name': name
                }).encode(), (peer['host'],peer['port']))
        except:
            print("Unable to reach peer: ", peer)
    last_flood_time = now
```

```
# Setting the target
t_stats = {}
for addr in stats:
    s = stats[addr]
    for t_id in s:
        if t_id in t_stats:
            t_stats[t_id] = s[t_id] + t_stats[t_id].copy()
        else:
            t_stats[t_id] = s[t_id]
c_stats = {}
for t_id in t_stats:
    counter = {}
    x= t_stats[t_id]
    for name,status,infohash in x:
        if name not in counter:
            counter[name] = 0
        if status:
            counter[name] += 1
    c_stats[t_id] = counter



targets = {}
for c_stat in c_stats:
    sorted_stats = sorted(c_stats[c_stat].items(), key=lambda x:x[1])
    if sorted_stats[0][1] > 3:
        continue

    targets[c_stat] = sorted_stats[0]

# handling socket request
read_sockets, write_sockets, error_sockets = select.select( inputs, inputs, [], 5)
for client in read_sockets:
```

```
# browser request
if client.getsockname()[1] == WEBSITE_PORT:
    if client is sock2:  # New Connection
        clientsock, clientaddr = client.accept()
        inputs.append(clientsock)
    else:  # Existing Connection
        data = client.recv(1024)
        url = data.decode()[4:].split(" ")[0]
        if url == "/":
            peers_table = "<thead><td>Who<td>Where<td>last_ping</thead>"
            for peer in peers:
                peers_table += f"""<tr><td>{peer['id']}</td><td>{peer['host']}:{peer['port']}</td>
                                <td>{int(now-peer['time'])} seconds ago</td></tr>"""
            collection_table = "<thead><td>Messages<td>Torrents<td>Hash</thead>"
            all_stats = "<h2>Stats</h2>"
            for addr in stats:
                stats_table = "<thead><td>Messages<td>Torrents<td>Hash</thead>"
                stat = stats[addr]
                for id in stat:
                    s = stat[id]
                    stats_table += f"<tr><td>{id}</td><td>{s}</td><tr>"
                all_stats += f'<h4>{addr}</h4><table border="1">{stats_table}</table>'
            # send HTTP response to the browser
            client.send(f"""HTTP/1.1 200 OK\nContent-Type: html\n\r\n\r\n
                    <head><meta http-equiv="refresh" content="3"></head>
                    <body>
                    <h1>Beacon Status</h1>
                    hosted on: {host}:{port}
                    <h2>Current peers</h2>
                    <table border="1">{peers_table}</table>
                    <h2>torrents to download</h2>
                    {targets}
```

```
                    <h3>Currently downloading</h3>{"downloading_queue"}
                    <h3>Downloading Overflow</h3>
                    <h2>The Collection</h2>
                    <table border="1">{collection_table}</table>
                    {all_stats}
                    <h4>Stats Counter</h4>{c_stats}
                    <style>table{{border-collapse:collapse}}thead{{text-align:center;font-weight:
bold;}}</style>
                """.encode())
                client.close()
                inputs.remove(client)
            else:
                st=[]
                for addr in stats:
                    stat = stats[addr]
                    host,port = addr
                    st.append({'host':host,'port':port,'stat':stat})

                kb = float(1024)
                mb = float(kb ** 2)
                gb = float(kb ** 3)
                m = psutil.virtual_memory()
                memTotal = int(m[0]/gb)
                memFree = int(m[1]/gb)
                memUsed = int(m[3]/gb)
                memPercent = int(memUsed/memTotal*100)
                core = os.cpu_count()
                CPU_percent = psutil.cpu_percent(0.05)
                speed = psutil.net_io_counters(pernic=False)
                psend = round(speed[2]/kb, 2)
                precv = round(speed[3]/kb, 2)
                client.send((f"HTTP/1.1 200 OK\nContent-Type:
```

```
application/json\nAccess-Control-Allow-Origin: *\n\r\n\r\n"+json.dumps({
                    'targets':targets,
                    'host':host,
                    'port':port,
                    'c_stats':c_stats,
                    'stats':st,
                    'peers':peers,
                    'collection_files': collection_files,
                    "CPU_core":core,
                    "memory":   memTotal,
                    "CPU_percent": CPU_percent,
                    "RAM_used": memUsed,
                    "RAM_total": memTotal,
                    "RAM_percent": memPercent,
                    "sent" : speed[0],
                    "packet_send"      : psend,
                    "packet_receive"   : precv,
                    'torrents':qb.torrents(),
                    'c_stats':c_stats,
                    'collection_files': collection_files,
                    'comments': None,
                    'messages': None
                }}).encode())
            client.close()
            inputs.remove(client)

    # udp beacon/peer request
    else:
        try:
            data, addr = client.recvfrom(5*1024)
        except:
            continue
```

```
print ('recv %r - %r\n\n' % addr, data)
req = json.loads(data.decode())

if req['type'] == 'FLOOD':
    addr_req = (str(req['host'])), int(req['port'])
    for peer in peers:
        # #print("peer",peer)
        if peer['host'] == req['host'] and peer['port'] == peer['port']: # old peer
            peer['time'] = time.time()  # update time for clean up timeout
            break
    else:  # new peer
        peers.append({
            'id': req['id'],
            'port': int(req['port']),
            'time': now,
            'host': str(req['host']),
            'name': req['name']
            })

    try:
        for peer in peers:
            client.sendto(data, (peer.host, peer.port))
    except:
        pass

    client.sendto(json.dumps({
        'type': 'FLOOD_REPLY',
        'host': host,
        'port': port,
        'name': name
        }).encode(), addr_req) # not replied to sender i.e. addr
```

```
elif req['type'] == 'STATS':
    client.sendto(json.dumps({
        "type": "STATS_REPLY",
        "data": torrents
        }).encode(),addr)

elif req['type'] == 'STATS_REPLY':
    stats[addr] = req['data']
    for stat in stats:
        if stat['addr'] == addr and stat['height'] == req['height'] and stat['hash'] == req['hash']:
            stat[time] = now
            break
    else:
        req.pop('type')
        req['addr'] = addr
        req['time'] = now
        stats.append(req)

elif req['type'] == 'GET_BLOCK':
    try:
        client.sendto(blockcollection.Blockcollection[req['height']].json().encode(),addr)
    except:
        client.sendto(json.dumps({
                'hash':None,
                'messages': None,
                'timestamp': None,
                'type': 'GET_BLOCK_REPLY'
            }).encode(),addr)
```