TEST Engineering & Management

September/ October 2020
ISSN: 0193-4120 Page No. 14-18

A LIGHTWEIGHT PEER TO PEER BACKUP SYSTEM FOR MULTIMEDIA DATA

Sangita Chaudhari[1], Amankumar Shrivastava[2], Siddharth Raja[3], Amulya Patil[4]

[1,2,3,4] Dept. of Computer Engineering, Ramrao Adik Institute of Technology, DY Patil Deemed to be University, Navi Mumbai, Nerul, Navi Mumbai, India.

Article Info
Volume 83
Page Number: 14-18
Publication Issue:
September/October 2020

Article History
Article Received*: 4 June 2020*
Revised*: 18 July 2020*
Accepted*: 20 August 2020*
Publication*: 15 September 2020*

Abstract

In the timeline of technological development, it will be fair to say that growth in the advancements of technology has sparked its consumption as well. With data being ever so more accessible, individuals consume huge amounts of data before they run out of functional storage space on their devices. A millennial kid would not believe the fact how commercialized Compact Discs were which offered only a few hundred megabytes. Similarly, soon the tons of storage that we seem to be familiar with now will be dwarfed by storage of a huge magnitude. Very soon individuals will find themselves often running out of primary storage on their device and will need additional space to accommodate new files without having to dump the existing files. With the recent trends and a surge in cloud computing services, it is only logical for data to be stored on the go on a cloud-based server. Our project aims to solve the same problem by creating a multimedia backup system where we make use of different technologies like Electron JS, Torrent Protocols to build a system that performs the task of intelligent multimedia backup where the files transferred will be deleted from the sender side to optimize storage.

1. Introduction

Data Backup owes its origin to saving important data from physical or virtual damage. The history of data backup starts with magnetic tapes which existed even before home computing was developed. Since the backup and retrieval process on tape was slow for business requirements, it was replaced by the disk drives developed by IBM. Although it was expensive it gradually established itself as a faster, efficient, and convenient way to back up the data. It gains popularity as it was very convenient to store and access data into it on personal computers. To save the data from physical damages they were stored in remote premises, safe in a dedicated business center. Fast Forward today cloud is the new and most efficient way to backup data remotely over the internet. The cloud is proven to extremely popular and efficient technology to store users' data securely with minimal pay peruse. It implements various algorithms and backend processes to efficiently store, transfer, and retrieve users' data. There are many players in the market like Amazon Web Services, Google Cloud, IBM, Microsoft Azure providing storage as a service.

Although such service providers are available, it does not support automatic backup of the data from users' machine as and when required. In a peer-to-peer system, such backup plays an important role at the time of node crashes. We aim to create a Device-to-Device backup system enabling users to store data on multiple devices. Besides, our system will provide an Automatic Deletion service that enables the user to delete after sending or uploading files. Often, we document files that might not be particularly useful sometime later thus eating up a significant amount of storage over time. A trivial example can be that every person has lots of images as memories, which are rarely viewed after a certain period of time. Now understand that in this kind of situation, the pictures occupy a huge chunk of memory and are rarely used. So, we planned accordingly to use secondary device storage to store this chunk of data with the secure transmission. Moreover, the product allows the user to retrieve the data once sent to the secondary device without requesting the receiver end. The product proposes a smart deletion

September/ October 2020
ISSN: 0193-4120 Page No. 14-18

system from the sender's device once data has been received by the receiver, the signal is sent to initiate a timer that automatically deletes files from the sender's system storage.

2 LITERATURE SURVEY

There are different ways in which backup and retrieval of real-time data can be done. Tilkov and Vinoski have proposed a solution to establish an asynchronous and high-performance network between peers for data sharing. It was observed that the system supports multiple browsers with a key focus on lesser memory consumption [1]. Pramukantoro et al. depict an interesting approach towards middleware in messaging systems. In IoT, multiple requests are made every second which eventually gave birth to a scalable middleware by designing a Cluster of Redis by implementing multiple protocols like COAP, MQTT. Once a server cluster is created, the transaction between the subscriber and its client was synchronized. [2]. Saini et al. illustrate the analysis of different computing and encryption approaches highlighting their space and time complexities apart from their compatibility with other components of the system. They have also studied major cloud computing players such as IBM, AWS, and Adobe and their performance for the storage of data [3]. Leang et al. proposed an integrated system of Apache KAFKA and spark Cluster for real-time data streaming and processing. Although it results in high-performance streaming but results in complex hardware requirements [4]. Gopal et al. suggested how the system load can be decreased by caching data objects from heavily loaded peers to other peers. It always maintains a threshold availability of data and enables dynamically sharing of files by calculating load at each node. [5]

While the existing systems do solve the problem of security, encryption, and efficient data transfer idiosyncratically, there are not many solutions that are robust in all the departments. Systems end up compromising on either extensibility or ease of use while building the approaches.

Most of the traditional multimedia backup systems work on the Client-Server model with specific servers and specific clients connected to the server. There is a client request for a service and the server responds with the service. The data is stored in the desired server. When several clients request services simultaneously, a server can get bottlenecked. The client-server is expensive to implement. These systems are less distributed in nature, moreover, real-time sharing of data is a point of concern for such systems.

3 PROPOSED SYSTEM

The system has been brought to function by using socket programming where the sender and the backup device are brought together in a secure and private channel where the eventual transfer of files will take place. The data produced on the sender's side is encrypted using standard Encryption Algorithms and then this data is shared using the torrent protocol. The torrent magnet link associated with the file is communicated to the backup device, wherein an automated function is triggered to download and store this file on the backup device. The second part of the system suggests a way of retrieving data from the backup device to the primary device. When the primary device requests the server for the file it is sent to the backup device with a specific filename, the cloud server request the backup device it is mapped to for the data it earlier received. Thus, the located file is traced back to the primary device. The data verification for every packet received by the backup device is done using Torrent Protocol which stores message digest(hash) using SHA- 256 which poses a compatibility challenge with the DHT and trackers, which have protocols that expect 20-byte hashes. To handle this, DHT- and tracker announcements and lookups for v2 torrents use the SHA-256 info-hash truncated to 20 bytes.

Traditional file-sharing systems do not share data in chunks, they are solely dependent on a centralized server where data is being served through REST endpoints using HTTP/HTTPS protocols. This system proposes a decentralized file-sharing mechanism using Web-Torrent APIs. Web Torrent is a streaming torrent client for the web browser and the desktop. Web Torrent is written completely in JavaScript – the language of the web – and uses WebRTC for peer-to-peer transport whenever possible. No browser plugins, extensions, or installation is required to use Web

September/ October 2020
ISSN: 0193-4120 Page No. 14-18

Torrent in your browser. The system is built on Electron.js which provides a headless Chromium browser to create a desktop client for file sharing/file backup. Currently, the system has Windows OS build associated with the client-side. Figure 3.1 shows the authentication and authorization system. Users are required to register themselves on our system with an email id and post successful verification they can begin their tasks, JWT tokens are used to verify the user credentials from the back end to maintain integrity as depicted in the figure above.

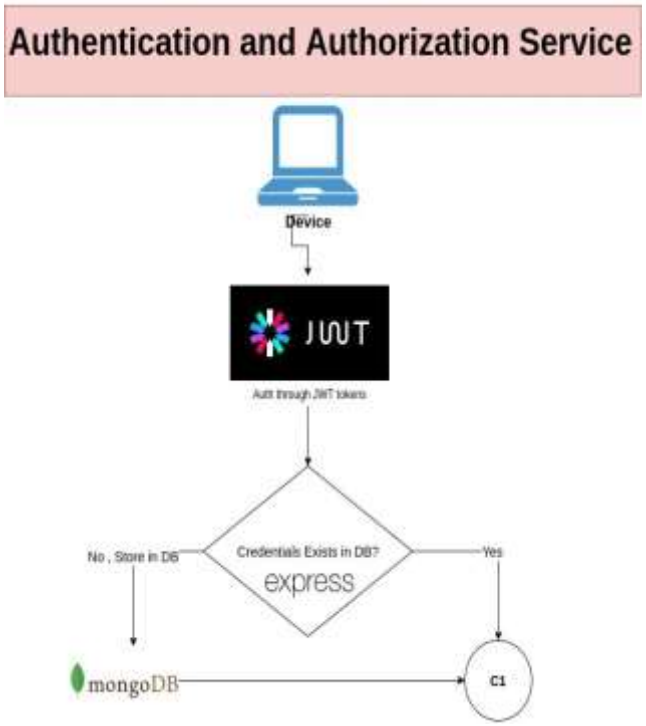

Figure 3.1: Authentication & Authorization

Figure 3.2 depicts the scenario of creating a grouping of nodes participating in the backup process. Users on participating nodes can create their own private rooms and can add other registered members to their room by sending them an invitation through mail. All the users in a room are connected to each other through a web-torrent protocol which enables the transfer of files between all the members where users are connected in a socket program. Figures 3.3 shows a fully connected network between all such participating nodes which enables efficient file transfer for backup and retrieval of the multimedia data over peers.

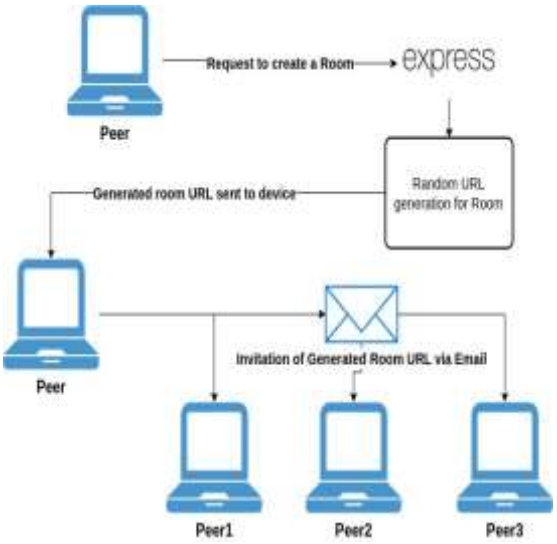

Figure 3.2: Room Creation & Invitation

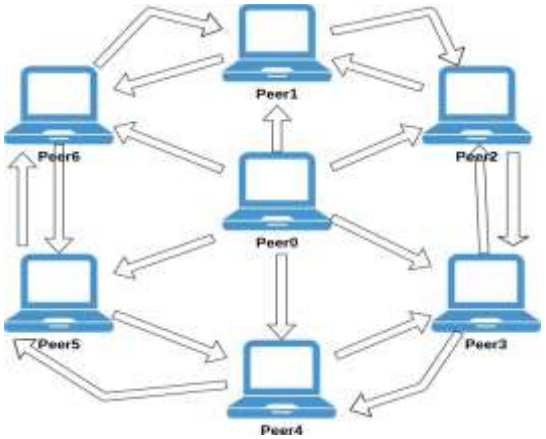

Figure 3.3: File Transfer in a Private Room

Figure 3.4 illustrates the retrieval of the backup files. For retrieval, the sender initiates a request to be sent to the express server. The express server informs the backup device about the same. It will be retrieved directly if its Magnet URL is already present in the cache otherwise it will be retrieved from the database. Torrent tracking has logs through which it sends the file back to the sender.

3.1 Backup Mechanism

Today in the modern era of the twenty-first century, the latest advancements in technology have empowered the user to produce data in bulk that does not fit in the local storage of the device they carry. People today produce tonnes of data over different web platforms/social media and even locally in their machines. Cloud Solutions today provide users extreme power to store,

September/ October 2020
ISSN: 0193-4120 Page No. 14-18

fetch, process and manipulate data in less time and efficiently. But these Cloud technologies do not provide an open source proposed system/methodology of how they tend to store data in the required format. Thus, this report proposes a system of Backing-up Data from one device to the other, to minimize dependency of Cloud-usage.

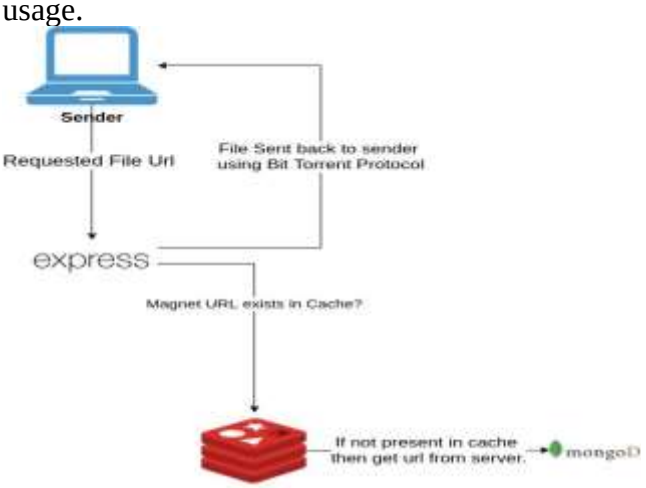

Figure 3.4: Data Retrieval Process

Figure 3.5 shows the detailed process of backup and retrieval of the data. It consists of the following steps:

1) Users acting as primary, or backup devices can register and login themselves. Proper emailing service is implemented for scenarios such as Forgot Password, Registration Success / Failure, and Invitation Mails for users to join private rooms.

2) Registered users can create a room where creators can invite any other user via their email-ids.

3) Once the socket program is created between the users connected in a room, the connected users can upload files that are sent to the Web-Torrent Third Party Package.

4) Magnet URL received on the sender side is broadcasted to other members in the group via socket program. The sender in this case acts as a seeder.

5) The shared Magnet URL is received by other members, and they start downloading files over their system after reading the file in a chunked buffered format.

6) Logs are maintained in both sender and receiver ends consisting of the filename, timestamp, and sender email id. To retrieve a file these logs are read on the sender's side and a retrieve button is shown.

7) Sender can click that button and pull the file from the other device by sending the file name shared by him earlier in the group.

8) The reverse process of sharing the file then occurs from the receiver's end to the sender's end via Web- Torrent using magnet URL.

9) Caching for the least recently used data for Room Mapping, User mapping, and User-Room mapping is done in Redis DB which provides numerous in memory data structures to store the data efficiently in the server end which can be read efficiently.

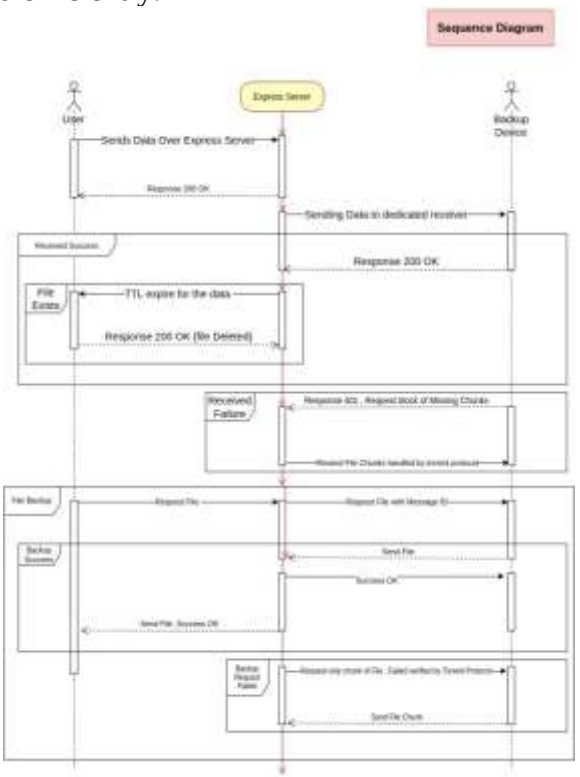

Figure 3.5: Backup and Retrieval Process

4 RESULTS AND DISCUSSION

This proposed system basically aims to build a decentralized multimedia backup system using Torrent protocol. The torrent protocol allows one to establish a peer-to-peer network between sender and receiver devices. The issue with the implementation of the naive torrent protocol is that it cannot be accessed over the web. This issue is resolved by Web-Torrent third-party package which develops a streaming torrent over the web. It uses WebRTC for peer-to-peer transport whenever possible. No browser plugins, extensions, or installation is required to use Web Torrent in your browser.

This system primarily focuses on bringing sender and receiver devices (acting as primary and backup devices) to connect over a common network and share

September/ October 2020
ISSN: 0193-4120 Page No. 14-18

or backup files using Web-Torrent. Once the user registered successfully on the service, he could make his own server rooms and add members to it. The upload button enables the user to send any file across the server to its members. The server log on the backup device shows the successful automated download of the files. The robustness of the system is demonstrated by the successful transfer of different types of files like .pdf, .png, .jpeg, .txt, .mp4, etc which can be seen in Table 4.1 Our system has no restrictions whatsoever when it comes to the file type and its size.

Table 4.1: File Transfer Statistics

File Size	File Type	Transfer Time
5MB	TXT	33s
10MB	PDF	54s
50MB	MP4	79s
100MB	BINARY	149s
250MB	EXE	308s
500MB	EXE	787s

5 CONCLUSIONS

Backup of the data and its retrieval is the crucial process in the peer-to-peer network while dealing with fault handling and its recovery process. A simple, yet a lightweight mechanism to deal with such a situation is needed to achieve a higher response time. Integrating the front end on the wheels of Electron JS and the NodeJS powered back end, we created a file-sharing system that uses the torrent protocol and socket programming to enable smooth multimedia transfers. The performance indices will be much higher in an ecosystem where advanced hardware and computing tools are easily accessible. Our system will need to undergo testing with extensive load before it can be really trusted for an enterprise or commercial release.

References

[1]. Tilkov, Stefan, and Steve Vinoski. "Node. js: Using JavaScript to build high-performance network programs." IEEE Internet Computing 14.6 (2010):
80-83.

[2]. Pramukantoro, Eko Sakti, et al. "A cluster message broker in IoT middleware using Ioredis." 2018 International Conference on Sustainable Information Engineering and Technology (SIET). IEEE, 2018.

[3]. Saini, Kavita, et al. "E2EE For Data Security For Hybrid Cloud Services: A Novel Approach." 2018 International Conference on Advances in Computing, Communication Control and Networking (ICACCCN). IEEE, 2018.

[4]. Leang, Bunrong, Rock-Won Kim, and Kwan-Hee Yoo. "Real-time transmission of secured plcs sensing data." 2018 IEEE International Conference on Internet of Things (iThings) and IEEE Green Computing and Communications (GreenCom) and IEEE Cyber, Physical and Social Computing (CPSCom) and IEEE Smart Data (SmartData). IEEE, 2018.

[5]. Gopal, S. Venu, N. Sambasiva Rao, and SK Lokesh Naik. "Dynamic sharing of files from disconnected nodes in peer-to-peer systems." 2016 International Conference on Electrical, Electronics, and Optimization Techniques (ICEEOT). IEEE, 2016.