# Secure and Decentralized Live Streaming using Blockchain and IPFS


Shachindra
Blockchain Architect
REVOtic Engineering
New Delhi, India
shachindra@revoticengineering.com

Sagar Ganiga
Application Developer
REVOtic Engineering
New Delhi, India
sagar@revoticengineering.com

Shreya Saha
Application Developer
REVOtic Engineering
New Delhi, India
shreya@revoticengineering.com

Anish Mishra
Infrastructure & Security Architect
REVOtic Engineering
New Delhi, India
anish@revoticengineering.com

Meit Maheshwari
Blockchain Developer
REVOtic Engineering
New Delhi, India
meit@revoticengineering.com

Gaurav Kumar
Full Stack Developer
REVOtic Engineering
New Delhi, India
gk@revoticengineering.com



***Abstract* — Decentralized cloud systems are proving to be much more advantageous than centralized cloud systems. They distribute power away from a central authority, cut down operation cost, have greater fault tolerance, fewer trust requirements between storage providers and data owners and is less prone to attacks. InterPlanetary File System, a protocol to create a content-addressable, peer-to-peer method of storing and sharing hypermedia in a distributed file system can revolutionize how we share the media content over the internet. We provide an overview of the current systems to stream media over the internet and describe various problems that these systems face with regards to media delivery, governance, and distribution. We exhibit, how with the help of IPFS, Blockchain based Smart Contracts and HTTP Live Streaming (HLS), it is possible to minimize, avoid and diminish the problems associated with the traditional media delivery system and how we can improve the overall efficiency of media delivery systems. We explain how the conventional framework of media delivery can be transformed by IPFS based delivery network supported by HLS streaming for all kinds of distribution model (live or on-demand). We also propose a novel method to decentralize the cloud storage system using a separate server and client-side applications.**



***Keywords—IPFS, Internet media delivery networks, HLS, Streaming with IPFS, Distributed Ledger Technologies***


## I. Introduction

Video streaming over the Internet is one of the most popular activities worldwide. Streaming media accounts for over two-thirds of all internet traffic and it is estimated to jump 82 percent by 2020. To handle this ever increasing traffic of media, the internet has matured over the last 3 decades and the current scenario demand one more revolutionary change in the backbone of the internet. Today's media delivery networks are experiencing problems related to limited bandwidth, quality of streaming, security and censorship concerns. In this paper, we explain how these issues can be addressed with the help of emerging technologies like IPFS supported by HLS.

## II. Evolution of Streaming Media Technologies

The vision of streaming media occurred at a point when primary multimedia technologies were prominent among desktop users. They ran unique software created to decompress and render these files on the screen. The initial and key annex of this prototype was the conviction of downloadable media on the Internet. Nevertheless, this was not an adequate action for consumers with scant amounts of storage, slow internet connection, and limited tolerance. This emphasized the creation of streaming media, a technology that facilitates the user to experience a media content on-the-fly over the internet.

This technological evolution includes the following standards

- HTTP-Based Streaming
- First Servers and Protocols for Streaming Media
- First Video Codecs for Streaming Media
- RealSystem G2
- Distributed Media Delivery Networks

## III. Current Scenario Of Streaming over Internet

In the last few years, video-supported applications, and especially video streaming applications, have become quite popular. Many providers started publishing their content such as news, series, and movies on their dedicated Web sites or for this purpose used dedicated sharing services such as YouTube, Hulu in the US, and many others.

In the above mentioned services, video streaming is usually based on the HTTP and TCP, and the video player is embedded in a web browser. TCP is currently the most widely used transport protocol on the Internet but it is commonly considered to be unsuitable for multimedia streaming. The HTTP and TCP are general purpose protocols and were not specifically designed or optimized for streaming media delivery. Bandwidth is precious. When we have a lot of people requesting the media over the internet, there is a lot of bandwidth that travels across the internet. These connections are uni-cast and they stream to each individual person. If there are 500 people watching, each person has to get a copy of that media. In this case, bandwidth increases with the number of people. For 500 people we need 500 times the bandwidth. To tackle this

issue, companies like YouTube are dependent on CDNs. Basically, they are fixing this issue by adding lot more of computers and by moving the distribution centers nearer to the mass users. To make matters more complicated, the difference between yearly price drop in storage (40%) and bandwidth (26%) is almost double. All this means that we will have more people, streaming more data on channels that are not scaling as fast. This creates congestion that at one point in time cannot be solved by just adding more hardware.

Imagine you are in a lecture together with 100 people and you all watch the same video. What happens is that this video has to be fetched from the closest nodes of Google, streamed to each of the attendee's laptops and repeated 100 times. Instead of students, who have an identical copy of the video, sharing it with each other, we propagate large amounts of data, long distances, multiple times. Inefficient, but this is how HTTP works and it is creating large congestion problems on the backbone of the Internet.

The internet is not 100% centralized since no single corporation owns the entire internet. But relatively few large corporations are responsible for hosting essential elements of what we consider the internet. This kind of system naturally has a single point of failure. This point of failure can be misused to disable access to the content for an entire country. Also, these traditional streaming services are under the authority of one or few big organizations. Quality of service and censorship rules are decided by them and the users need to follow that without any questions. Making it a centralized system of power and authority. Current cloud architectures are completely reliant on large storage providers such as Google, Amazon or Microsoft acting as trusted third parties that store and transfer data. Encryption is not widely adopted and the current architecture is susceptible to many security vulnerabilities. Many storage devices rely on the same infrastructure which is why failures are often correlated across systems or files.

Decentralization can be defined as distributing power away from a central authority or location. In a decentralized storage network, the stored information is distributed across decentralized clients, and each client node encrypts the data to ensure data security and data integrity is maintained using a proof of retrievability. Putting data on an open peer to peer market drives down cost for storage devices. The data is also resistant to unauthorized access, tampering, censorship and data failures. A decentralized storage network has providers of storage capacity that are economically rewarded from the users renting storage using smart contracts. Decentralized storage networks allow for micropayments where payments are directly tied to audits of how the files have been stored. This minimizes how much trust is needed between storage providers and data owners. Decentralized networks cannot be controlled by one authority figure or government, but they are logically centralized as in there is one commonly agreed state and the entire system behaves like one supercomputer.

## IV. Proposed Solution

Our system uses the following components of modern tech to build a safe and efficient method for media streaming over the internet.

### *A. InterPlanetary File System (IPFS)*

IPFS or InterPlanetary File System is an open-source protocol and network designed to create a content-addressable, peer-to-peer method of storing and sharing hypermedia in a distributed file system. It aims to make the web faster, safer, and more open. IPFS works by connecting all devices on the network to the same file structure. This file structure is a Merkle DAG, which combines Merkle trees, and Directed Acyclic Graphs (used in Git version control, which also allows users to see the versions of content on IPFS).

This is the process of adding and retrieving files from IPFS.

1) Each file and all of the blocks within it are given a unique fingerprint called a cryptographic hash.
2) IPFS removes duplicates across the network.
3) Each network node stores only content it is interested in, and some indexing information that helps figure out who is storing what.
4) When looking up files, you're asking the network to find nodes storing the content behind a unique hash.
5) Every file can be found by human-readable names using a decentralized naming system called IPNS.

HTTP is inefficient and expensive. HTTP downloads a file from a single computer at a time, instead of getting pieces from multiple computers simultaneously. With video delivery, a P2P approach could save 60% in bandwidth costs. IPFS makes it possible to distribute high volumes of data with high efficiency. And zero duplication means savings in storage. IPFS keeps every version of your files and makes it simple to set up resilient networks for mirroring of data. The webs centralization limits the opportunity. IPFS aims to replace HTTP. IPFS is becoming a new major subsystem of the internet. If built right, it could complement or replace HTTP. In IPFS streaming we don't need to push the content to every user. All you have to do is, push the content to the IPFS gateways. Anyone who wants that content can pick it up from there. IPFS gateway caches the content locally. So a number of gateways, more content sources. Each file and all of the blocks within it are given a unique fingerprint called a cryptographic hash. IPFS removes duplication across the network. Each network node stores only content it is interested in, and some indexing information that helps figure out who is storing what. Each network node stores only content it is interested in, and some indexing information that helps figure out who is storing what. When looking up files, you're asking the network to find nodes storing the content behind a unique hash.

IPFS provides persistence storage of those live streamed media content so that if someone misses the live streaming, they can always come back to find a saved copy of it. The network automatically deletes duplicates and tracks version

history. With our interface we also allow users to schedule the content for live streaming and broadcasting. Users can record the media and automate the system to stream that media on the given time for the specified duration. IPFS helps to resolve congestion and overly controlling governments by distribution. Instead of locations, IPFS addresses point directly to the resources and it makes sure that this data comes from the closest sources. This means that if a classroom full of students would watch the same video, they would fetch it from each other instead of any central location. This would make streaming a 4k video bufferless.

Another advantage of IPFS is that the user can download parts of a file from various sources at once and combine it at their side rather than downloading the whole file from a single source.

### B. *HTTP Live Streaming (HLS)*

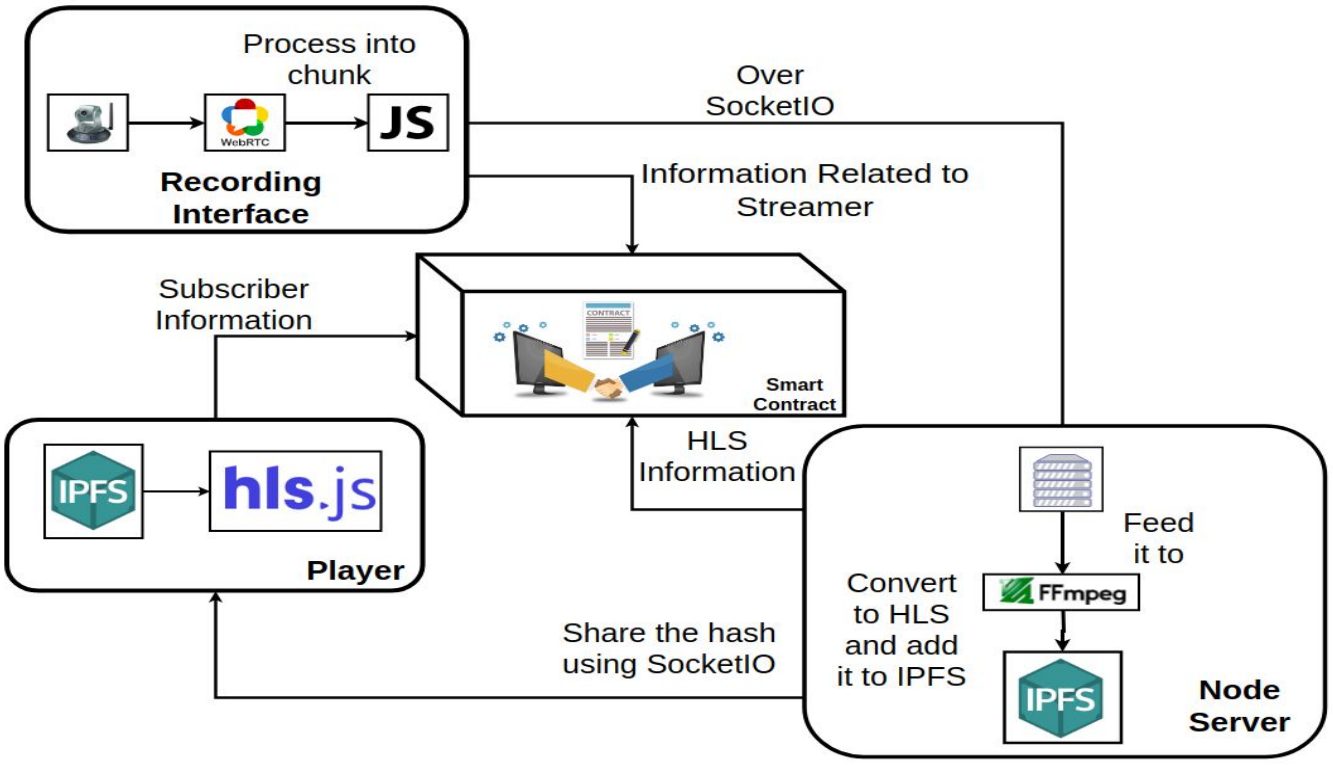
Streaming performance over IPFS can be increased by optimizing and compressing the video in right formats. HLS or HTTP Live Streaming is a video streaming format first introduced by Apple. It breaks streams into small file-based segments which are widely supported format for viewing streams in almost real time. HLS is designed for reliability and dynamically adapts to network conditions by optimizing playback. HLS can be easily integrated with HTML 5. HLS is quickly making its way up the technology ladder, thanks to its superior features and versatility. HLS divides the video chunk into fragments of equal length, kept as .ts files. It also creates an index file that contains references of the fragmented files, saved as .M3U8.

When an HLS video stream is initiated, the first file to download is the manifest. This file has the extension M3U8 and provides the video player with information about the various bit rates available for streaming.

### C. *How IPFS Helps in Streaming*

There are 2 parts to this system, one who publishes the data and one who subscribes. We are calling the publisher section as "medeia_publisher" and subscriber as "media_subscriber". The architecture below describes how each component works and communicates with each other.

1) Getting the Content
   a) WebRTC provides web browsers and mobile applications with real-time communication via simple application programming interfaces.
   b) With WebRTC we transfer the webcam content to node server as WebM with VP9 codec every 10 seconds.
2) Transferring this data to Server
   a) From Publisher, we send this WebM data over Socket.io to our server
   b) We create different room/folder for each session with the ID of Publisher.
   c) While doing this, Publishers can select the target node they want to save this data. Either they can store the chunks on their local gateway or on the cloud-based gateway.
3) Converting to HLS
   a) Once we receive the content chunk at server side, using FFMPEG we convert that chunk into HLS format.
   b) It creates an m3u8 file and ts files. We add these files into a folder specifically dedicated to this chunk inside the folder dedicated for this session of the live stream.
4) Adding the files to IPFS



a) We add the newly created chunk folder to IPFS.
b) In this way, we only add a small chunk of data to IPFS every 10 seconds and send this hash of it to subscribed viewers

5) Sending this data to subscribers
   a) Subscribers receive the hash for chunk as soon as it is added to the IPFS
   b) This transfer of chunk happens over Socket.io
   c) Each session has a different socket session and this hash is shared over that session.
6) Play it on the Subscriber Side
   a) As we get the hash on the subscriber side, HLS.js picks hash from the queue and starts playing it.
   b) HLS.js is connected to IPFS and gets the data for the given hash directly.
7) The stored version of the live session
   a) Once the live session is over, the publisher has the ability to add the recorded version of this live session to IPFS and share its hash over the socket.
   b) This will have the entire session recorded during the live session.
8) Pin the content locally
   a) We provide a separate server and client-side API (discussed in more detail in I) to pin the media content locally to the local IPFS node of the user for faster and efficient access.
   b) We are able to remove files from the main server that is already being hosted by a large number of clients. Hence, the storage is not wasted.
   c) This also leads to decentralization of the entire system.

This is one cycle of chunk recording, conversion to HLS, adding to IPFS, transfer to the subscriber and playing it on HLS.js. This operation happens continuously for various concurrent live streams.

### *D. FFMPEG*

We are using FFMPEG to convert the incoming chunks into HLS. FFMPEG is a free and open-source utility for converting, recording, splicing, editing, playing, encoding, muxing, demuxing, and streaming multimedia files. It can work with audio, images, and video in basically any codec or format used in the past 20 years. With FFMPEG, we can convert between different file formats and codecs, adjust bitrate (both audio and video) and broadcast a live stream video feed. It can effortlessly parse a file to any format, convert it to a different format and even transmit it through a network via any protocol. Storing incoming chunk on SSD will further improve the performance and conversion time.

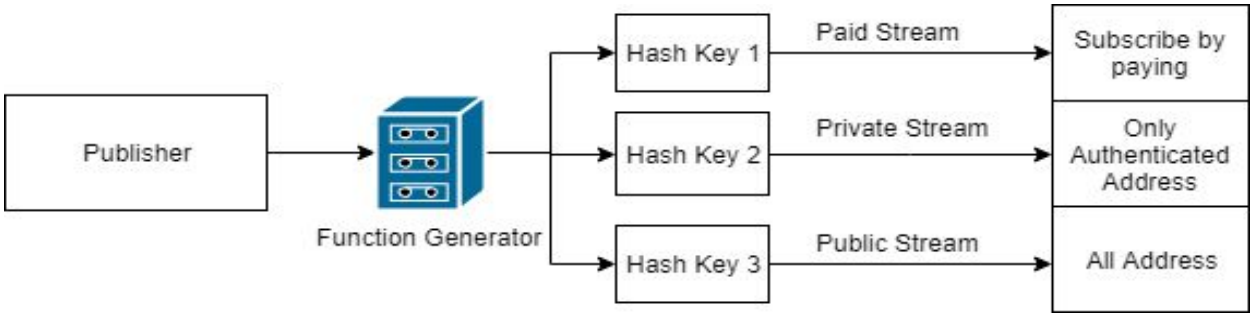
### *E. Smart Contract*

1) Authentication

With the increasing threat to the data leak in the current architecture. Data reliability is 0%. As anyone can manipulate personal data and steal or pretend to be that person.

   a) With the help of blockchain, we can authenticate whether user claims to be is the same person or not.
   b) Manipulating the data or reviewing someone else data, is next to impossible. As the only user has access to those personal data.
   c) As mentioned above, it becomes easy to authenticate the ownership of the streaming data. And it cannot be changed at any given point in the system
   d) This authenticity is achieved, in the system with the help of blockchain. As two address cannot be the same in the system. These nature of the blockchain provide the authenticity of the streaming data in IPFS.

2) Authorization
   a) After the streaming gets complete over the blockchain, a file hash is generated and stored in the blockchain mapped with the publisher address. As this process will help the users to authorize and authenticate the use of the streaming data by other users.
   b) Streamer can put the validation as to who can view or download the data. This process puts the extra layer of security on the data streamed over the IPFS network.
   c) Three types of function generator paid streaming, private streaming, and public streaming. All the functionality have a different way of working. Please find the brief details of the same below:
      i) Paid Streaming: User will have to pay tokens to watch the streaming
      ii) Private Streaming: User will invite the other users or a group
      iii) Public Streaming: Every user can view the streaming without spending tokens



d) As mentioned above, it becomes easy to authenticate the ownership of the streaming. And it can not be changed by anyone. This authenticity is achieved, as no two address can be the same in the system.

3) Accessibility
   a) Being the nature of decentralization, Blockchain offers a wide range of accessibility to the data.
   b) The publisher can set the access level to the data added in the IPFS node, knowing the level and importance of the data.
   c) As blockchain is easily accessible by connecting the node. With the inclusion of IPFS, it becomes very easy to access the data even without internet.
   d) Whenever system, is connected to internet and node gets sync with the network, Data's that are accessible can be view at any given point of time.
   e) After adding the accessible level, the publisher can also validate whether the same person has accessed the data or not.
4) Security
   a) Data leak, data hack, duplication, and many other threat levels which today's generation facing, the integrity of data is lost.
   b) Security to the data added in any web application can be stolen at any given point making it vulnerable.
   c) Blockchain uses many different algorithms to create the hash. And all the data is stored in the hash. It is practically impossible to crack the hash.
   d) Once the transaction is done in the blockchain, and validators, validate the data. It becomes next to impossible to break the security. Making data very secure and reliable.
   e) The source of the data is always known and it helps to track the authenticity of the data. Fake data or spam data can be blocked at any given time if the regulation is followed.
   f) Connecting the IPFS node to blockchain makes it very secure, as no one can steal the data. The only authorized person can view or edit the data.
5) Token Transaction
   a) Uploading data into IPFS or streaming data using IPFS can be used to generate the revenue for both publisher or viewer
   b) The publisher can set the particular token amount to view or subscribe to the data. Making it easy to earn medium
   c) Transaction of token takes very less amount of time as blockchain uses the concept of remittance free world.

## F. *Use Cases*

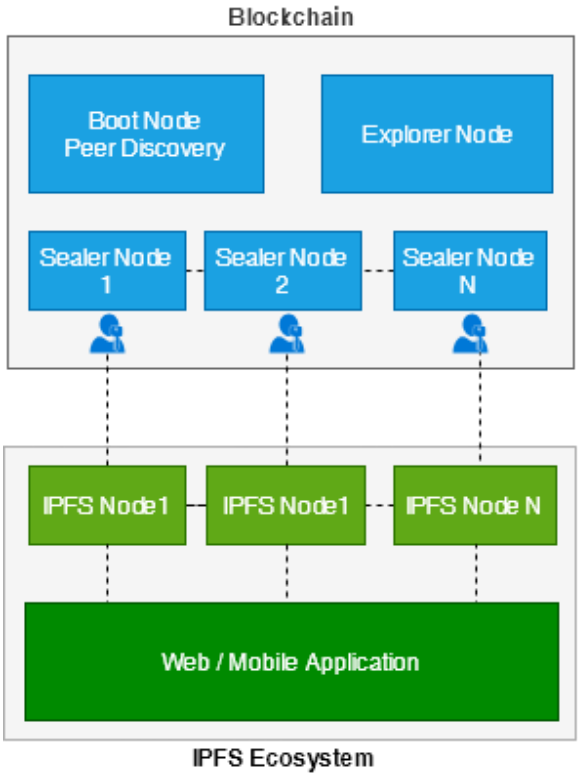
1) Protected one to one private live video chat
2) Secure live stream broadcasts
3) Secure and controlled Chat Rooms
4) Decentralized Television Network
5) Decentralized subscription based on-demand video
6) The decentralized and secure surveillance system

## G. *Scalability & Reliability*

Above proposed solution is scalable due to the nature of IPFS and private POA blockchain that is being used. Having private blockchain in place gives us easy access to scale our this platform up to 120 transactions per second.

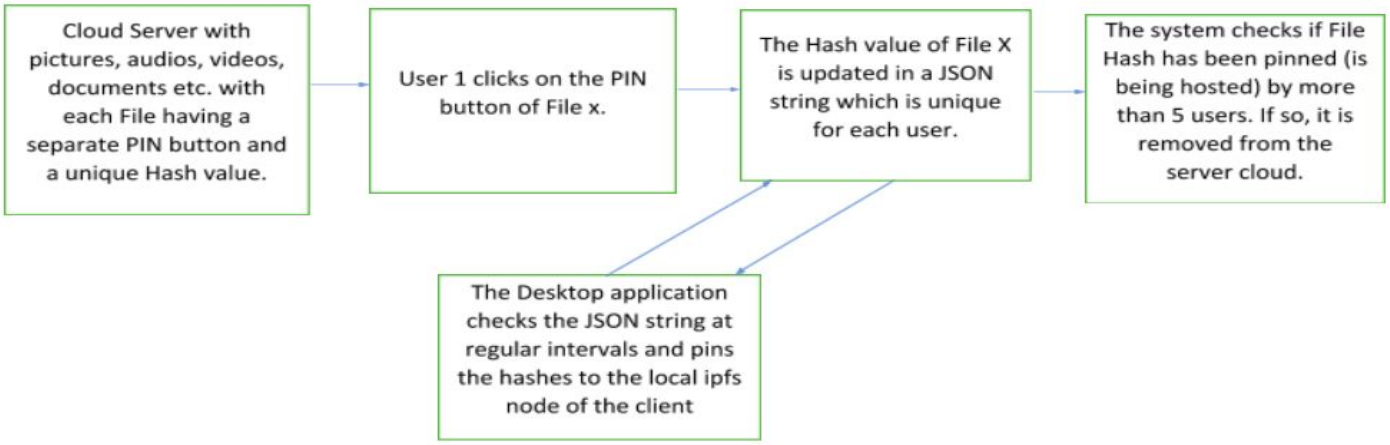
The native web application that gives access to the platform could be highly distributed and decentralized by enabling it to be part of IPFS itself.
A lightweight application can sit on the client side to serve dynamic content as well. This enables us to scale and support an increasing number of concurrent live streams.

Having Distributed Ledger Technologies and IPFS along with lightweight client-side application installed by the user can benefit in terms of performance and reliability as well.




## H. *Security*

May it be peer to peer stream or broadcast, it is highly secure and streamer only can control who can have access to their stream(s). Each live stream sessions are recorded on blockchain using unique session token and is controlled by specialized smart contract and it holds all necessary metadata to identify and associate the stream with streamer and viewers.




Streamer can allow specific viewers/wallet address on blockchain and authorize them to view the stream while viewers with valid wallet & private key can only subscribe to the stream and have access to the stream. This approach is highly secure & reliable harnessing the power of decentralization.

This gives tremendous flexibility to control the accessibility of data or stream allowing streamer to be the only identity-defining, managing and maintaining the authorization and security. Only the owner of data can define and control the security parameters associated with their stream, which is way more practical, reliable and secure compared to other means of centralized data delivery.

### *I. Proposal to decentralize Cloud Storage System*

We create a cloud server which stores pictures, audios, videos, and documents. Each of these files is added to IPFS and are assigned a unique hash value. The user has the option to pin any of these files. JSON arrays are maintained unique to each user which contain the hashes of the file pinned by each user, i.e., whenever a user pins a file, its hash value is added to the respective user's JSON array. A desktop application is running on the machines of each user, which checks the JSON array of the user at regular intervals and pins the file hashes to the local IPFS node of the user. When more than a certain number of clients pin the same file to their local node, it is removed from the cloud-based IPFS node, making its primary use as an IPFS Gateway.

### A. Desktop Application

This is in the form of a web app running locally on the user's machine. The pre-requirement of the application is that ipfs.exe should be preinstalled on the client's computer. This application dynamically pins files that the client wants to host from a server cloud to the local IPFS node of the client. The server-less cloud maintains a list of the file hashes that a client wants to host in the form of a JSON string which is updated regularly. The desktop application checks the JSON string at regular intervals and pins files to the local IPFS node of the client. The desktop application also has features where the user can start the IPFS daemon, check if the daemon is running or not and manually pin files to the local IPFS node.

Incentive Model: The users who pin the content locally on their computers are rewarded the token based on certain factors as the duration of storage (i.e. pinning of the hash), size of storage, bandwidth etc. This helps strengthen the IPFS network and allows the users to get an incentive for renting out their hard disk drive.

This model is similar to Filecoin except the fact that the platform decides which users would store specific files and issue tokens periodically.

### B. Web Application

A lightweight, interactive and secure web application will be used to provide native functionality of authorizing users. A new user can create their account to use IPFS streaming services. They can monitor the stats of IPFS network. Create API key to communicate between a client application and streaming server. Over web application, a user can also check documentation and guides to effectively use IPFS streaming technology.

### C. API Server

It primarily performs two major functions. It creates a blockchain wallet for a user taking username and password as the input. A public-private key pair is created for every username. The public key is returned to the user and the private key is encrypted with the entered password and stored. It is also responsible for adding files to IPFS and pinning them to the server IPFS node. The files are categorized into images, audios, videos, and documents. The API also maintains a record using JSON arrays (for every user) that contains the files hashes pinned by each individual user. A file is unpinned from the server IPFS node in case it is pinned by six or more users to their local systems. It basically is a lightweight, interactive and secure web application that will be used to provide functionality to the users.

## REFERENCES


[1] Gregory J. Conklin, Gary S. Greenbaum, Karl O. Lillevold, Alan F. Lippman, Yuriy A. Reznik, "Video Coding for Streaming Media Delivery on the Internet", IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY, VOL. 11, NO. 3, MARCH 2001

[2] Hareesh.K, Manjaiah D.H, "PEER-TO-PEER LIVE STREAMING AND VIDEO ON DEMAND DESIGN ISSUES AND ITS CHALLENGES", International Journal of Peer to Peer Networks (IJP2P) Vol.2, No.4, October 2011

[3] Tomas Kupka, "On the HTTP segment streaming potentials and performance improvements.", Faculty of Mathematics and Natural Sciences at the University of Oslo, February 2013

[4] Biernacki, A. & Tutschku, K. Multimed Tools Appl (2014) 72: 1143. https://doi.org/10.1007/s11042-013-1424-x

[5] Amber Case, "Why The Internet Needs IPFS Before It's Too Late", TechCrunch Article, October 2015

[6] Rohith Gandhi, "InterPlanetary File System(IPFS) — Future of the Web", CoinMonk, Medium Article, May 2017

[7] Prasenjit Chakraborty, Sachin Dev, Rajaram Hanumantacharya Naganur, "Dynamic HTTP Live Streaming Method for Live Feeds", 978-1-5090-0076-0/15, DOI 10.1109/CICN.2015.333© 2015 IEEE, pp.1394-1398

[8] R. Pantos, W. May and Apple Inc. Internet draft - https://tools.ietf.org/html/draft-pantos-http-live-streaming-23

[9] Smita R Gupta, Krunal Panchal, "MODELING REAL-TIME MULTIMEDIA STREAMING USING HLS PROTOCOL", IJARIIE-ISSN(O)-2395-4396, Vol-2 Issue-6 2016

[10] Yang Can, Li Yongyan, "A New Mobile Streaming System Base-on Http Live Streaming Protocol", 978-1- 4244-6252-0/11©2011 IEEE

[11] Ankit Songara1 and Lokesh Chouhan, ``Blockchain: A Decentralized Technique for Securing Internet of Things" in International Conference on Emerging Trends in Engineering Innovations \& Technology Management (ICET: EITM-2017).

[12] Marcella Atzori, 'Blockchain Technology and Decentralized Governance: Is the state still necessary?' in Journal of Governance Regulation/ Volume 6, issue, 1 2017

[13] Protocol Labs, 'Filecoin: A Decentralized Storage Network', July 19, 2017

[14] Juan Benet, David Dalrymple, Nicola Greco, 'Proof of Replication', Protocol Labs, July 27, 2017