DecenTube: A decentralized video streaming platform (dApp) using Ethereum and IPFS

Hemendra M. Naik†, Shubham A. Vaity and Shannon N. D'mello
Students, B.E. Computer Engineering, Xavier Institute of Engineering, University of Mumbai, Mumbai, Maharashtra, India
Saurabh D. Patil
Assistant Professor, Department of Computer Engineering, Xavier Institute of Engineering, University of Mumbai, Mumbai, Maharashtra, India

Video streaming platforms such as YouTube, Vimeo, and Dailymotion are both a boon and bane in our lives, as this privilege comes bundled with the responsibility of user privacy and data. Video streaming platforms store data in a server that is vulnerable to crashes. Decentralized applications, on the other hand utilize blockchain technology, which is a series of blocks connected sequentially. These transactions remain immutable. Thus security is enhanced. Ethereum based applications are decentralized. Distributed storage reduces the cost of server-side hardware and increases data availability. IPFS (Interplanetary file system), a peer to peer hypermedia protocol, stores immutable data, removes duplication and obtains address information for storage nodes to search for files in the network. We have also used a collaborative IPFS cluster along with the traditional IPFS system. There is a crypto-incentive system which rewards the uploader with ERC-20 (Ethereum Request for Comments) tokens.

Keywords: Smart Contracts, IPFS, Remix IDE, Video Streaming, Ethereum, Vyper, Ganache, Truffle, MetaMask

1. INTRODUCTION

In recent times more and more internet surfers are using digital content or subscription-based online services for music and video streaming (Chavan, Warke, Ghuge, and Deolekar, 2019). Digital content provides mobility and easy access. These conventional online streaming platforms are beset with drawbacks such as cost, piracy and lack of privacy. These disadvantages can be overcome with IPFS and Ethereum. Internet users do not possess control over their data, which they share today on websites. Ethereum is unique. It attempts to lever the blockchain to correct a problematic segment of the internet's design. Decentralization of an application makes the application more efficient, scalable, and fault-tolerant. A smart contract combined with IPFS will be used. IPFS is a distributed technology which is more economical than blockchain storage.

2. LITERATURE REVIEW

2.1 Decentralized System Architecture

A decentralized system is a network where nodes do not depend on a single master node. Control is distributed among many nodes. Cryptocurrency service providers such as Bitcoin, Ethereum, Litecoin, etc. use this model of delivering services.

Some examples of decentralized systems are DSound and DLive. DSound is a decentralized platform, which allows a user to upload, listen to and discover music and other sounds stored on top of the IPFS Network (Dsound, 2020). DLive is a live streaming video service constructed on top of the STEEM blockchain (DLive, 2020).

†Corresponding Author

2.2 Objectives of Decentralized System

- Decentralized video platforms empower the participants, allowing the market to make direct connections towards pricing, revenue distribution, content selection, advertising model, etc.
- A truly decentralized infrastructure is unyielding to attempts at shut down, there being no single point of instability.

3. EXISTING SYSTEM

3.1 Centralized System Architecture

Currently, a centralized system is in use for video uploading and sharing. Centralized systems are conventional (client-server) IT systems wherein a single authority controls the system and is solely in charge of all its operations. Users of a centralized system depend on a single source of a service. Online service providers, such as eBay, Google, Amazon and others use this common model of delivery of services.

Examples of centralized systems are Dailymotion, Vimeo etc. Dailymotion is a French video-sharing technology platform primarily owned by Vivendi. Vimeo is an ad-free video streaming platform providing free video viewing services. Figure 1 depicts the existing CDN system.

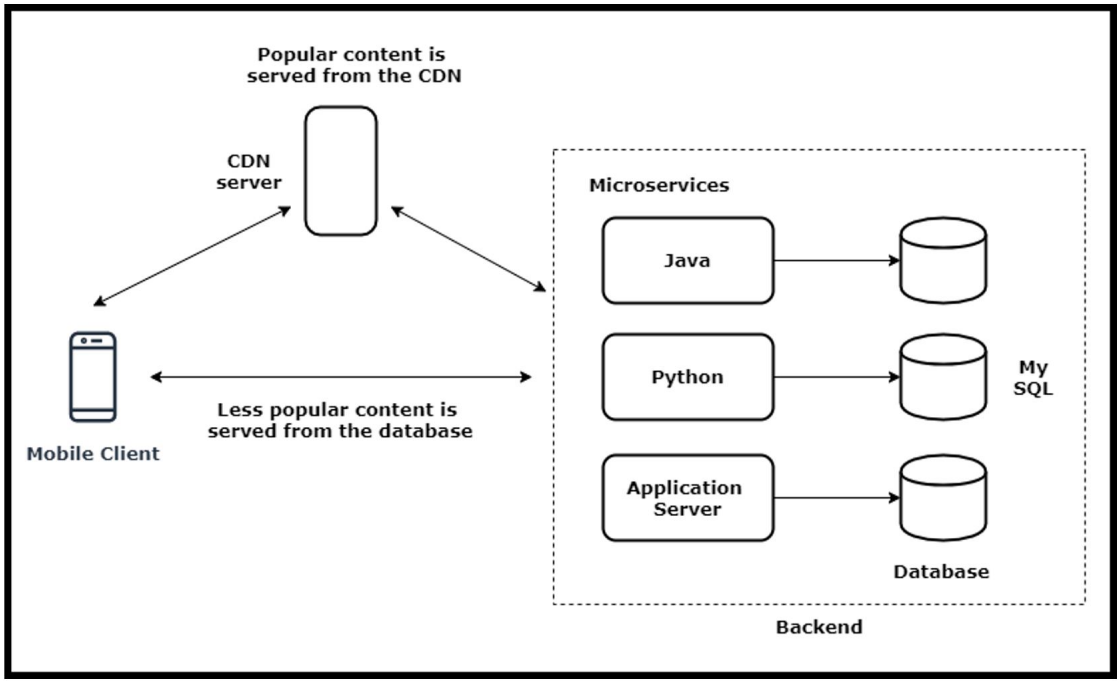


Figure 1. Existing System: CDN.

3.2 Objectives of Centralized System

- Revenue models of these systems depend mainly on advertisements.
- The ensuing problems with these services are multiple: eg. end-users are bombarded with ads, content creators get unfair revenue share, inefficiencies with content delivery and promotion of binge consumption without providing true-value content (Sathish, Patankar, and Khanna, 2019).

3.3 Traditional video storing methods

Most of these systems depend on AWS or other cloud services for storage of videos. If the company does not approve of an uploaded video, they can immediately take it down, and the URL that is present in the smart contract becomes invalid. Since storage of data is costly on the blockchain, so only the IPFS hash value of the video will be stored on the blockchain.

4. PROPOSED SYSTEM

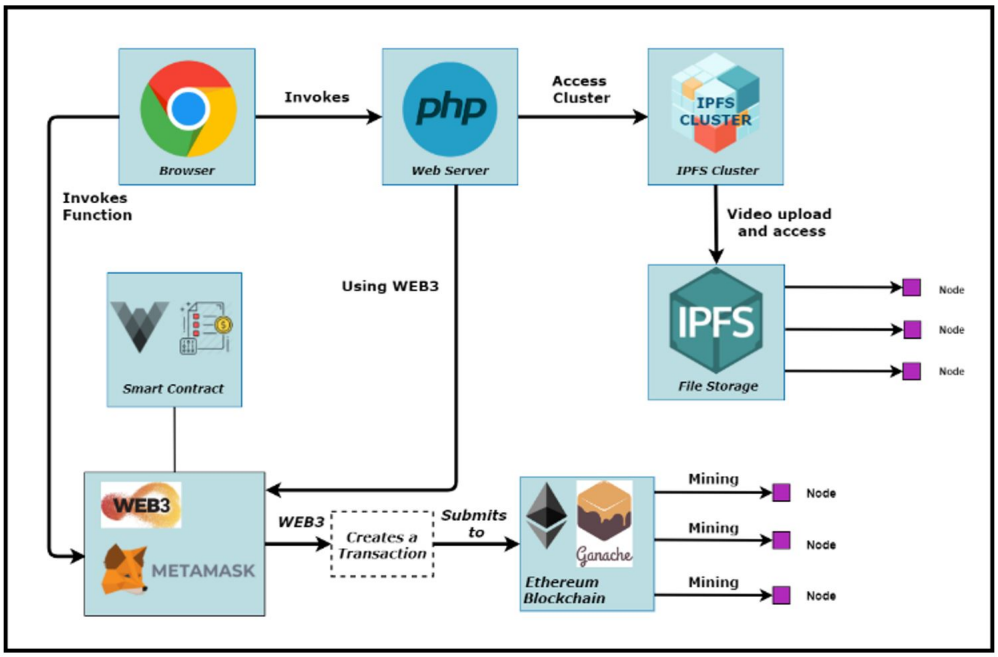


Figure 2. Block diagram of decentralized video streaming application.

Figure 2 represents the block diagram of our decentralized video streaming application. A web browser with the addition of MetaMask extension (Chrome and Firefox) becomes a dApp (Decentralized application) browser (MetaMask, 2020). These browsers provide an interface to interact with a dApp.

A PHP server along with MySQL database is used for basic processing and to store general user data and video metadata. The user can interact with the dApp for performing actions like watching, uploading videos and liking the videos. While performing actions like liking and uploading, the number of likes and other related metadata is stored in the Ethereum blockchain.

IPFS, a distributed file system has been used to store the videos. It is a P2P protocol. Every file stored on an IPFS node has a unique hash value (IPFS, 2020). Along with IPFS we are using a collaborative IPFS Cluster which provides data backup and distribution across a swarm of IPFS daemons (IPFS-Cluster, 2020).

5. DATABASE DESIGN

Our video streaming platform makes use of the Ethereum blockchain to maintain public record of transactions. Although blockchain serves as the best source of trust, the website holds a MySQL database for efficient data processing (Le, Kim, and Jo, 2019). Use of MySQL database provides a means for video search on DecenTube. Figure 3 depicts the ER diagram of the system.

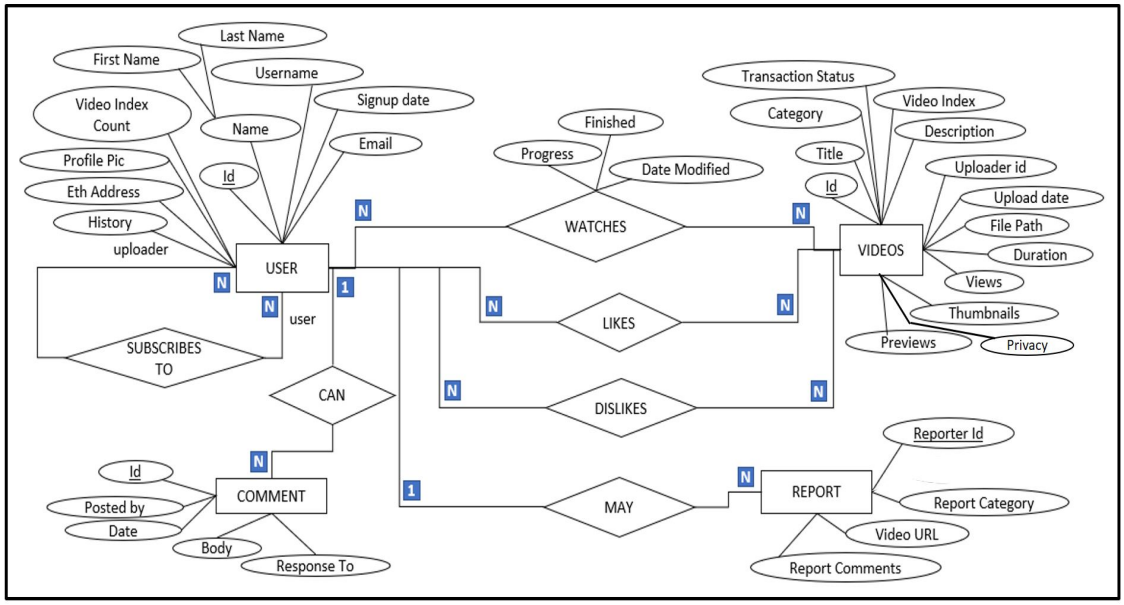


Figure 3. Entity-relationship diagram of our system.

The entities are as follows:

- User: All user related metadata along with subscriber and uploader details are stored in this entity.
- Videos: All video related metadata is stored in this entity.
- Comment: This entity is used to store details about the comments on the video or the replies to a comment.
- Report: It stores details about the reported videos.

6. SMART CONTRACT DEVELOPMENT

The DecenTube smart contract has been successfully deployed to the Ropsten Test Network via the Remix IDE (Integrated Development Environment). Remix is an IDE that is used to write, compile, debug and deploy Solidity and Vyper smart contracts (Remix, 2020). We have written our smart contracts using Vyper. Vyper is a contract-oriented, pythonic programming language (Vyper, 2020). Vyper makes it harder for developers to deliberately write misleading or malicious code and also protects developers from unintentionally leaving vulnerabilities in their smart contracts (Kaleem and Laszka, 2020). There is one primary struct which our smart contract maintains i.e. Video. Figure 4 represents the structure of the Video struct.

```
struct Video:
    path: bytes[50]
    title: bytes[200]
```

Figure 4. Video struct.

The video struct contains the IPFS hash value of the video and the title of the video (Kok, 2019).

We are using DTC (DecenTube Coin) tokens to reward the uploaders. These are ERC-20 tokens. Ergo we have implemented the following methods (Vogelsteller and Buterin, 2015):

- totalSupply()
- balanceOf()
- transfer()
- transferFrom()
- approve()
- allowance()

In addition, we have implemented methods for liking/disliking videos, subscribing/unsubscribing to channels, buying/selling DTC tokens etc.

6.1 Function to like videos

```
@public
def like_video(_user: address, _index: uint256) -> bool:
  _msg_sender_str: bytes32 = convert(msg.sender, bytes32)
  _user_str: bytes32 = convert(_user, bytes32)
  _index_str: bytes32 = convert(_index, bytes32)
  _key: bytes[100] = concat(_msg_sender_str, _user_str, _index_str)
  _likes_key: bytes[100] = concat(_user_str, _index_str)

  assert _index < self.user_videos_index[_user]
  if self.likes_videos[_key] == False:
        if self.dislikes_videos[_key] == True:
                self.dislikes_videos[_key] = False
                self.aggregate_dislikes[_likes_key] -= 1
        self.likes_videos[_key] = True
        self.aggregate_likes[_likes_key] += 1
        self._transfer(msg.sender, _user, 4)
        self._transfer(msg.sender, self.admin, 1)
        log.LikeVideo(msg.sender, _user, _index)
        return True
  elif self.likes_videos[_key] == True:
         self.likes_videos[_key] = False
        self.aggregate_likes[_likes_key] -= 1
        log.UnlikeVideo(msg.sender, _user, _index)
        return True
  return False
```

Figure 5. Code snippet for liking a video.

Figure 5 depicts the code for liking a video. The like feature toggles between like and un-like of the video. Once a user likes a video, he confers a reward of 5 DTC tokens. Out of these 5 tokens, 4 are transferred to the video uploaders ethereum wallet and 1 is transferred to DecenTubes administrators ethereum wallet towards maintenance.

6.2 Purchase of DTC tokens

```
@public
@payable
def buyTokens (_numberOfTokens: uint256) -> bool:
  assert(block.timestamp > self.lastTokenBuyTime[msg.sender] + 604800)
  assert(msg.value == self.multiply(_numberOfTokens, self.tokenBuyPrice))
  assert(_numberOfTokens <= 1000)
  assert(self.balances[self.admin] >= _numberOfTokens)
  self._transfer(self.admin, msg.sender, _numberOfTokens)
  self.tokensSold += _numberOfTokens
  self.lastTokenBuyTime[msg.sender] = block.timestamp
  log.BuyTokens(msg.sender, _numberOfTokens)
  return True
```

Figure 6. Code snippet for buying ERC-20 tokens.

We have set up an ICO (Initial Coin Offering). An ICO is the cryptocurrency industry's equivalent to an IPO (Initial Public Offering). Users can purchase DTC tokens. These tokens can be used on our platform. Every user can buy tokens only once per week and a maximum of 1000 DTC tokens per transaction. Figure 6 represents the smart contract code to purchase DTC tokens.

7. VIDEO ENCODING

FFmpeg is a complete, cross-platform solution to record, convert and stream audio and video. FFmpeg has been used for encoding videos and generating thumbnails & GIFs. All uploaded videos are encoded into the MP4 format with 480p quality before adding them to the IPFS network, since MP4 is a widely used extension which is compatible with many web video players.

8. WORKING

8.1 Working of video uploading feature

Figure 7 represents the sequence diagram for uploading videos. The user initiates the process by uploading a video. The server encodes the video as well as generates a thumbnail and a GIF preview using FFmpeg.
After that the IPFS node will compute the hash value of the rendered video and send it back to the server. Then the server will send a MetaMask transaction confirmation pop-up to the user. After successful transaction, the video metadata is stored in the MySQL database and the smart contract.

8.2 Working of video liking/disliking feature

Figure 8 depicts the sequence diagram for displaying and liking the video. The user selects a video which he desires to watch. The server will fetch the hash value of this video from the smart contract and accordingly the video is fetched from the IPFS network.

If a user likes a video, he will click on the like button, and this like will be stored in the smart contract. As soon as the video is liked, the uploader will be rewarded with ERC tokens.

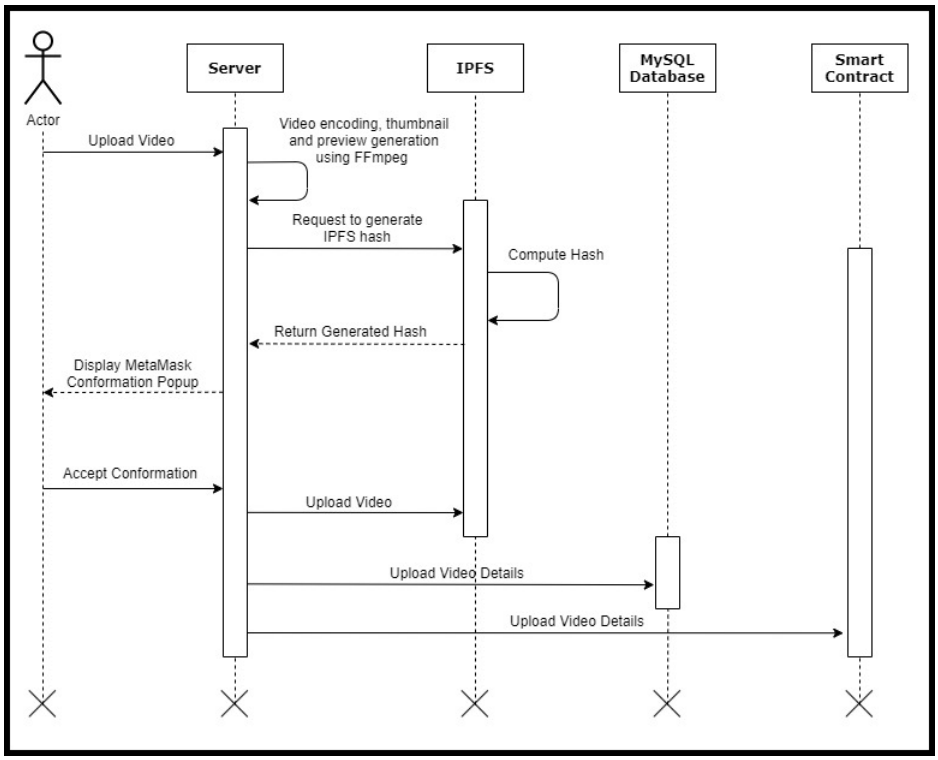


Figure 7. Sequence diagram for uploading a video.

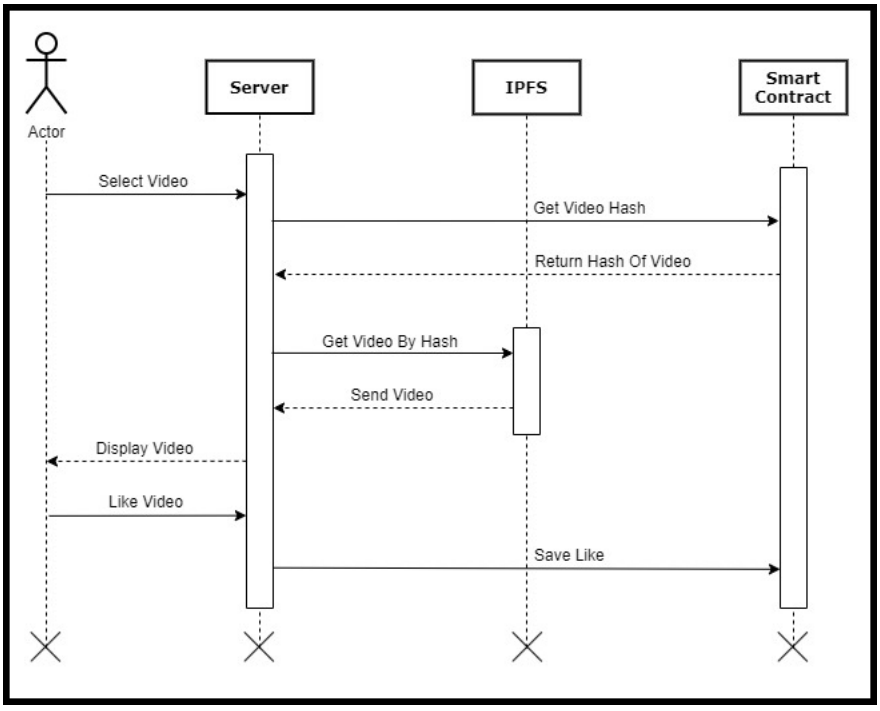


Figure 8. Sequence diagram for liking/disliking a video.

9. WEB INTERFACE

We have developed a website for DecenTube, which has been made very user friendly. The website has the following features:

- Like/Dislike videos (shown in Figure 9 & 10)
- Option to choose an IPFS gateway to stream from (shown in Figure 11)
- Upload videos (shown in Figure 12)
- ICO sale page (To purchase DTC Tokens) (shown in Figure 13)
- Subscription Feed (shown in Figure 14)

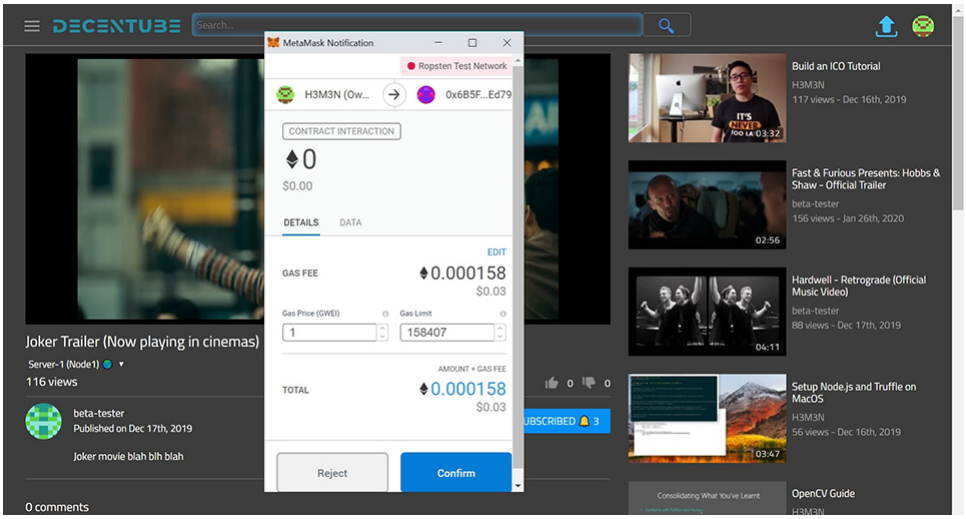


Figure 9. MetaMask confirmation for liking a video.

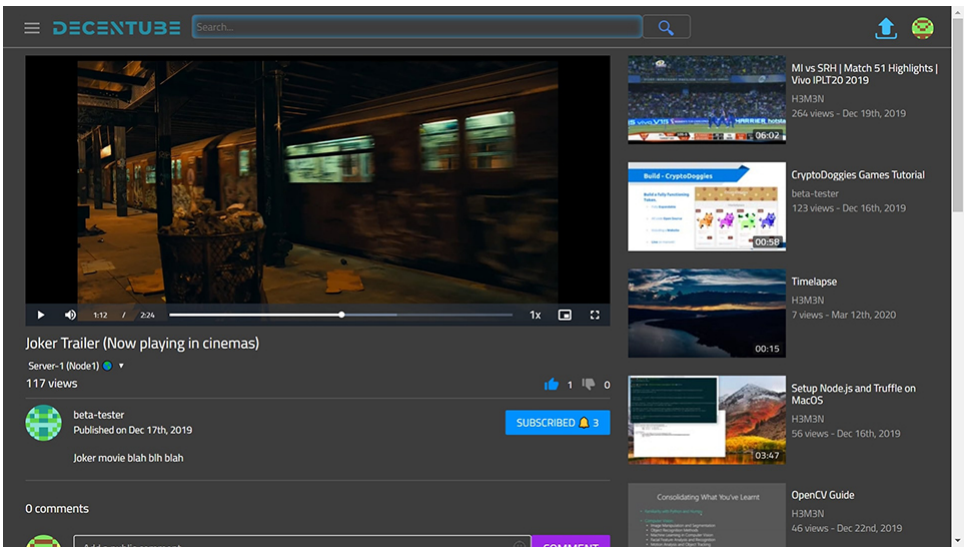


Figure 10. The like button gets highlighted after video is liked.

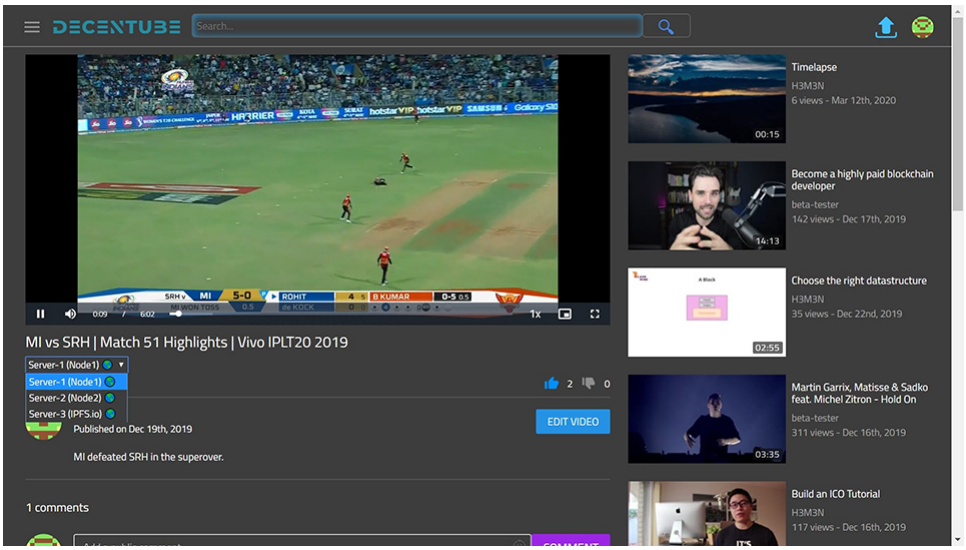


Figure 11. Selecting the IPFS node to stream from.

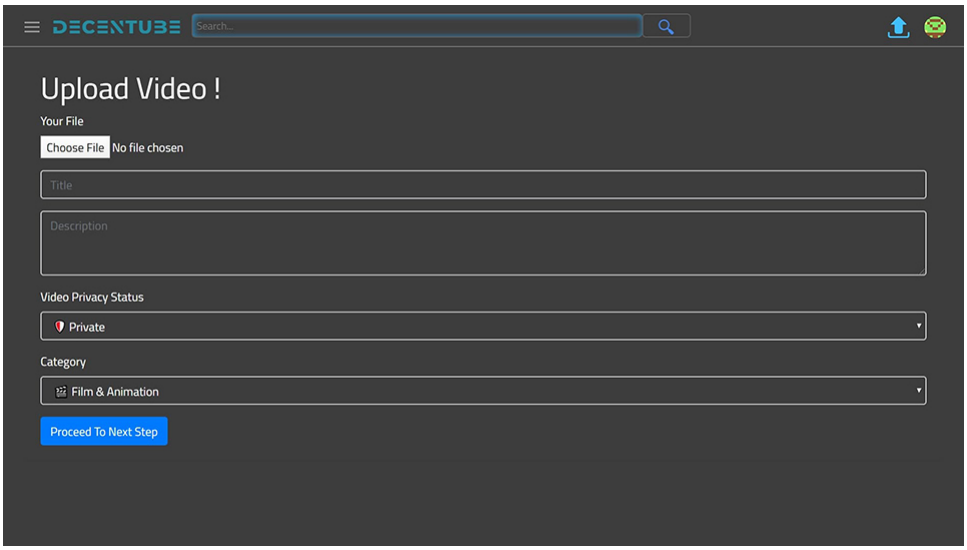


Figure 12. Upload video page.

10. EXPERIMENT RESULTS AND DISCUSSION

DecenTube DApp was analyzed by running experiments over the Ropsten Testnet (Ethereums Test Network) (DecenTube-Smart-Contract, 2020). The results are shown in Table.I. As of 1st May 2020, 1 Ether equals 16,173.21 Indian Rupee (Google, 2020).

11. SMART CONTRACT TESTING

Using JavaScript, tests can be run on the smart contract to ensure whether it is functioning properly (Le et al., 2019).
Figure 15 shows the section of code used to test the creation of DTC token. The function waits for the contract to be deployed then gets the information of the ERC-20 token.

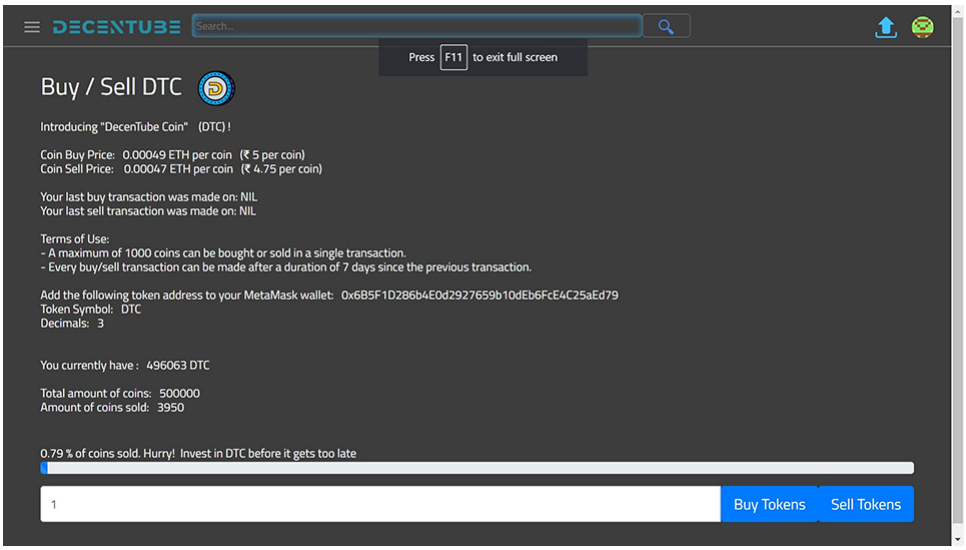


Figure 13. ICO sale page.

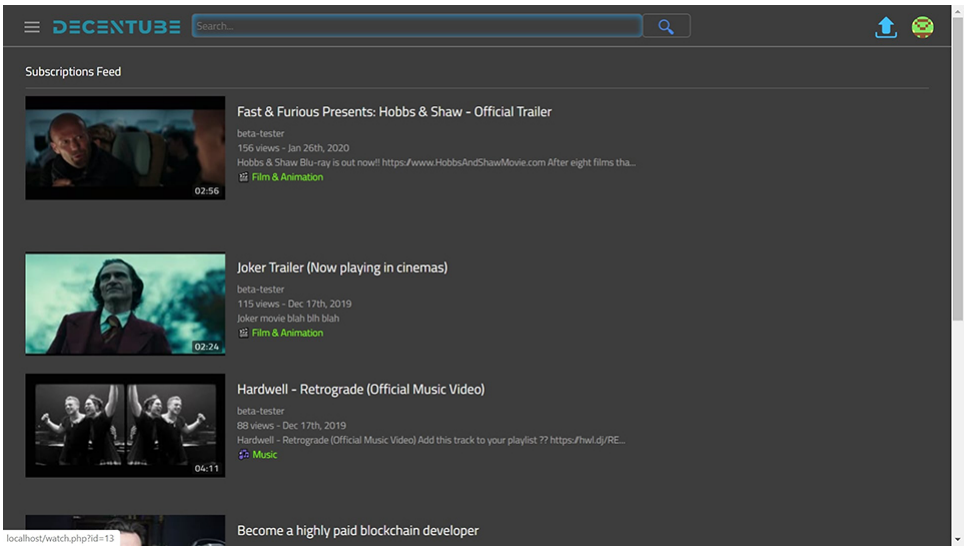


Figure 14. Subscription feed.

12. SMART CONTRACT VULNERABILITY ANALYSIS

Smart Check is an open source tool that allows users to audit their smart contract and highlights the vulnerabilities in the code and gives a detailed explanation of the problem (SmartCheck, 2020). We have analyzed our smart contract for security vulnerabilities.*

13. CONCLUSION

Our decentralized application (dApp) running on IPFS and blockchain confers advantages and scalability. Decentralization and distribution are the milestones achieved. Traditional solutions make use of centralized servers to store data. This raises the storage cost due to several servers, regular maintenance checks, and administrative costs.

*The smart contract audit report can be accessed at https://tool.smartdec.net/scan/1a100de9b2194f0896552ff19239f57c

Transaction / Function	**Gas Price (Gwei)**	**Gas used by transaction (GAS)**	**Transaction Fee (Ether)**	**Transaction Fee (INR)**
VideosSharing	1	2,024,499	0.002024499	32.74
Like	1	105,605	0.000105605	1.71
Unlike	1	20,211	0.000020211	0.33
Dislike	1	84,028	0.000084028	1.36
Undislike	1	20,225	0.000020225	0.33
Subscribe	1	84,385	0.000084385	1.36
Unsubscribe	1	23,294	0.000023294	0.38
Upload Video	0.000166705	149,051	0.000000024	0.00040
Rename Video	1	74,545	0.000074545	1.21
buyTokens	2	104,415	0.00020883	3.38
sellTokens	1	80,310	0.00008031	1.30
withdrawBalance	1	31,054	0.000031054	0.50

Table I: Costs incurred by performing transactions on Ropsten Testnet Network.

```
const VideosSharing = artifacts.require("VideosSharing");
require('chai')
  .use(require('chai-as-promised'))
  .should()
contract('VideosSharing', ([deployer, user_acc1, user_acc2,
user_acc3]) => {
  let videossharing
  before (async () => {
    videossharing = await VideosSharing.deployed()
  })
  describe('Deployment and Initialization of contract:', async ()
=> {
    it('Deploys successfully', async () => {
      const address = await videossharing.address
      assert.notEqual(address, 0x0)
      assert.notEqual(address, '')
      assert.notEqual(address, null)
      assert.notEqual(address, undefined)
    })
    it('Has correct amount of totalSupply: 500000', async () =>
{
      const totalSupply = await videossharing.totalSupply()
      assert.equal(totalSupply.toNumber(), 500000)
    })
    it('Deployer has entire supply of tokens', async () => {
      const deployer_balance = await
videossharing.balanceOf(deployer)
      assert.equal(deployer_balance.toNumber(), 500000)
    })
    it('has correct name: Video Sharing Coin', async () => {
      const name = await videossharing.name()
      assert.equal(web3.utils.toAscii(name), 'Video Sharing
Coin')
    })
    it('Has correct symbol: VID', async () => {
      const symbol = await videossharing.symbol()
      assert.equal(web3.utils.toAscii(symbol), 'VID')
    })
    it('has correct number of decimals: 3', async () => {
      const decimals = await videossharing.decimals()
      assert.equal(decimals.toNumber(), 3)
    })
  })
})
```

Figure 15. Code snippet to test the created ERC-20 token.

Thus, we have developed a decentralized video streaming application where files are stored in a distributed manner using IPFS. We have also setup a collaborative IPFS cluster. Users are given a choice to join and distribute content. This makes our system more scalable and efficient. We have also devised a crypto-incentive system by rewarding the uploader with ERC-20 tokens based on the videos popularity (number of likes). In this paper, we have provided an inexpensive alternative for a centralized video sharing platform.

References

Chavan, S., Warke, P., Ghuge, S., and Deolekar, R. V. 2019. Music streaming application using blockchain. In *6th International Conference on Computing for Sustainable Global Development (INDIACom)*. New Delhi, India, pp.1035–1040.

DecenTube-Smart-Contract. 2020. Ropsten testnet network - etherscan. `https://ropsten.etherscan.io/address/0x6b5f1d286b4e0d2927659b10deb6fce4c25aed79` (accessed April 29, 2020).

DLive. 2020. Dlive - live stream on blockchain. `https://dlive.tv/m/home/` (accessed April 24, 2020).

Dsound. 2020. Dsound - decentralized sound platform. `https://dsound.audio/` (accessed April 25, 2020).

Google. 2020. Eth to inr, google search. `https://www.google.com/search?q=eth+to+inr` (accessed May 01, 2020).

IPFS. 2020. Interplanetary file system. `https://ipfs.io/` (accessed May 04, 2020).

IPFS-Cluster. 2020. Ipfs cluster- pinset orchestration for ipfs. `https://cluster.ipfs.io/` (accessed May 04, 2020).

Kaleem, M. and Laszka, A. 2020. Vyper: A security comparison with solidity based on common vulnerabilities. ArXiv, abs/2003.07435.

Kok, A. S. 2019. Implementing a decentralized application using ipfs. In *Hands-on Blockchain for Python Developers: Gain Blockchain Programming Skills to Build Decentralized Applications Using Python*. Packt Publishing, p.382.

Le, T., Kim, Y., and Jo, J. 2019. Implementation of a blockchain-based event reselling system. In *6th International Conference on Computational Science/Intelligence and Applied Informatics (CSII)*. Honolulu, HI, USA, pp.50–55.

MetaMask. 2020. Metamask - a crypto wallet & gateway to blockchain apps. `https://metamask.io/` (accessed March 03, 2020).

Remix. 2020. Remix - ethereum ide. `https://remix.ethereum.org/` (accessed February 26, 2020).

Sathish, S. K., Patankar, A. A., and Khanna, H. 2019. Aurum: A blockchain based decentralized video streaming platform. In *IEEE Wireless Communications and Networking Conference (WCNC)*. Marrakesh, Morocco, pp.1–8.

SmartCheck. 2020. `https://tool.smartdec.net/` (accessed February 04, 2020).

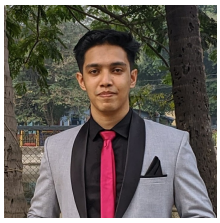
Vogelsteller, F. and Buterin, V. 2015. Eip 20: Erc-20 token standard. `https://eips.ethereum.org/EIPS/eip-20` (accessed January 18, 2020).

Vyper. 2020. Vyper - contract-oriented, pythonic programming language. `https://vyper.readthedocs.io/en/latest/` (accessed April 15, 2020).

Hemendra M. Naik is a graduate student in the Computer Engineering program at Xavier Institute of Engineering (XIE), Mumbai, India. He has a background in Web Development. He holds keen interests in the area of blockchain and P2P networks. He is well-versed in technology and writing code to create systems that are reliable and user-friendly. He was leading the Website Development Team for the annual inter-college festival Spandan 2019 at XIE, Mumbai. He can be reached at hemendranaik@gmail.com

Shubham A. Vaity is an undergrad student in the Department of Computer Engineering at the Xavier Institute of Engineering (XIE), Mumbai, India. He has experience in Web Development. He has an interest in Data Science and Machine Learning. He has worked on Mini-projects based on Data Mining, Image Processing, and Blockchain Technology. He was the volunteer at the Website Committee for the annual inter-college festival Spandan 2019 at XIE, Mumbai. He can be reached at vaity.shubham@gmail.com

Shannon N. D'mello is an Computer Engineer graduate from Xavier's Institute Of Engineering, Mahim. He is good in coding and has working background in Web Development. D'mello has strong interest in field of Blockchain. Python is his preferred language for coding. He can be reached at shannondmello@gmail.com



Saurabh D. Patil is an Assistant Professor in Xavier Institute of Engineering, Mumbai, India. He is currently working on a vehicular network for designing framework emergency management. He has done his master's project on security protocol design for VANETs. He has subject expertise in Computer Networks, Compilers and Wireless Communication. He also has over 10 years experience in academics. He can be reached at saurabh.p@xavier.ac.in