Decentralized Social Media Platform

Design and Implementation of a Censorship-Resistant Social Media Application using IPFS and Ethereum

CENG 3550: Decentralized Systems and Applications

Tuna Güven (230709064)
Esin Güler (230709075)

Department of Computer Engineering

January 20, 2026

Abstract

This report details the design and implementation of a decentralized social media application (DApp) developed as a final project for the Decentralized Systems course. The project addresses the limitations of traditional Web 2.0 social platforms, specifically focusing on censorship resistance, data ownership, and fair revenue distribution for creators. The system utilizes a hybrid architecture combining the Ethereum blockchain for logic and transaction settlement, and IPFS (InterPlanetary File System) for distributed content storage. Key features include Self-Sovereign Identity (SSI), an NFT-based marketplace, encrypted content delivery, and a zero-fee tipping mechanism. We demonstrate data consistency and network persistence through a custom Host-Client topology using Tailscale and the Go implementation of IPFS (Kubo).

1 Introduction

Traditional social media platforms rely on centralized servers, creating single points of failure and giving platform owners absolute control over user data. This centralization leads to issues regarding censorship, privacy violations, and unfair monetization models where intermediaries extract significant fees from content creators.

We propose a Web 3.0 alternative: a Decentralized Social Media DApp. By leveraging blockchain technology and distributed file systems, our application ensures that users retain ownership of their content and identity. The platform is built upon the Ethereum Hardhat environment for smart contract execution and IPFS for storage, ensuring censorship resistance. Furthermore, we implement a fair economic model where creators receive 100% of the tips (likes) directly from their audience, eliminating platform fees.

2 System Architecture

The application follows a hybrid decentralized architecture. To overcome the high cost of storing large media files on the blockchain, we utilize IPFS for heavy static assets (images, audio, video) while using the Ethereum blockchain for state management, identity verification, and financial transactions.

2.1 Cooperative Pinning Protocol

The core innovation of our architecture is the "View-to-Host" protocol, which ensures data persistence through user participation. Figure 1 details the sequence of events between a Host (User A) and a Guest (User B) connected via a secure Tailscale mesh network.

1. **Initial Upload & Discovery:** The process begins when the Host (User A) uploads a file (e.g., "Hello.png"). This file is initially stored *only* on User A's local IPFS node. When the Guest (User B) opens the application, their client queries the Ethereum Blockchain to retrieve the post list and obtains the unique Content Identifier (CID).

2. **P2P Data Transfer:** User B's client requests the content using the CID. Since the file is not yet on User B's machine, their IPFS node uses the Swarm protocol to locate User A and download the data directly via the Tailscale tunnel.

3. **Auto-Pinning (The Innovation):** Once User B successfully views the content, the application logic automatically triggers an `ipfs.pin.add` command on User B's local node. This transitions User B from a passive "viewer" to an active "co-host," permanently saving the file to their local storage.

4. **Fault Tolerance (The "Unplug" Test):** To validate the architecture, the system is tested against a failure scenario where User A goes offline. As shown in the diagram, when User B refreshes the page, the content does not disappear. Instead, the local IPFS node retrieves the image from its own cache. This proves that data has successfully migrated and is no longer dependent on the original creator.

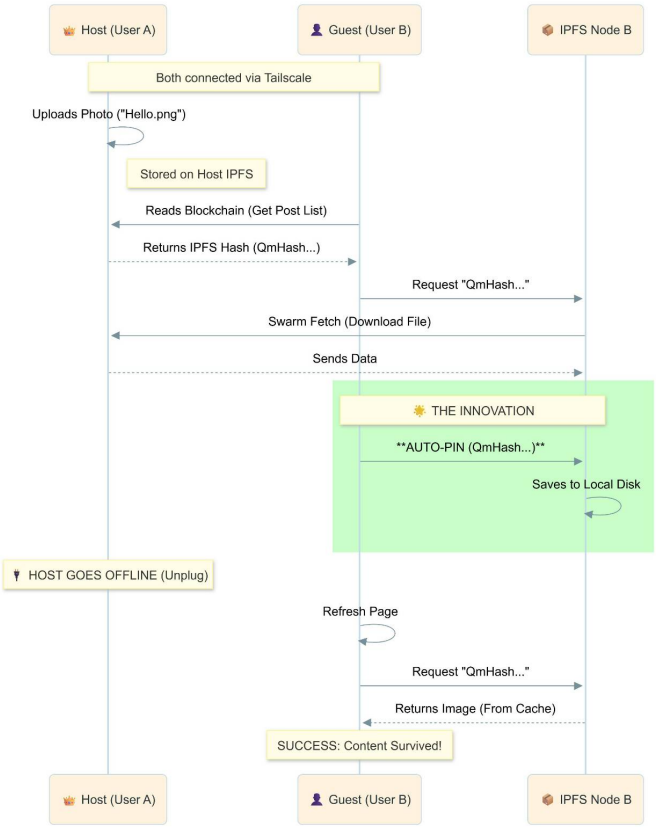

Figure 1: Sequence Diagram of the Cooperative Pinning Protocol, illustrating how data survives when the Host goes offline.

3 Technical Implementation

The core technology stack consists of **Kubo** (the Go implementation of IPFS) for storage nodes and **Hardhat** for the local Ethereum development environment. The frontend interacts with these layers via **Metamask** and Web3 libraries.

3.1 Wallet Integration and Authentication

Unlike traditional username/password schemes, our application uses cryptographic key pairs for authentication. Users log in using the **Metamask** browser extension. The user's Ethereum wallet address serves as their unique identifier within the system. For demonstration purposes, we utilized test accounts imported via private keys into Metamask to simulate various network actors.

3.2 Self-Sovereign Identities (SSI)

We implemented a Self-Sovereign Identity system to establish trust and authenticity. User profiles (Bio, Display Name, Avatar CIDs) are not stored on a central database but are recorded directly on the blockchain smart contract.

- This ensures that a user's identity is portable and immutable.
- Users can mathematically prove they are the owners of their profiles by signing transactions with their private keys.
- This creates a "Root of Trust" directly on the ledger.

3.3 NFT Minting and Marketplace

Creators can tokenize their content (music, paintings, digital art) by minting them as Non-Fungible Tokens (NFTs).

1. The media file is uploaded to IPFS, generating a Content Identifier (CID).
2. A smart contract function mints a token linked to this CID.
3. Creators can list these NFTs for sale at a desired ETH price.
4. Ownership is tracked on-chain, allowing for a decentralized secondary market.

3.4 Encrypted Posts and Token-Gating

A key feature of our platform is the "Subscribers Only" content model, achieved through encryption and NFTs.

The Encryption Logic: To ensure that only paying subscribers can view exclusive content, we implemented the following logic:

1. **Encryption:** When an artist posts premium content, the media file is encrypted client-side using a symmetric key (AES) before being uploaded to IPFS. This ensures that even if the IPFS CID is public, the raw content remains unreadable.

2. **Token-Gating:** The decryption key is not stored openly. Access is governed by the Smart Contract. The frontend application checks the user's wallet balance for the specific NFT associated with the post.

3. **Decryption:** If the user owns the required NFT (Proof of Payment), the system retrieves the encrypted payload from IPFS and allows the local wallet to decrypt the content for viewing.

3.5 Zero-Fee Tipping System

We replaced the traditional "Like" button with a financial transaction. In our system, a "Like" represents a direct monetary tip.

- **Mechanism:** When a user clicks "Like," Metamask prompts a transaction of **0.01 ETH**.

- **Direct Transfer:** The smart contract routes these funds directly from the Liker's wallet to the Content Creator's wallet.

- **Fair Economy:** The protocol charges **0% fees**, ensuring the artist receives the full value of the support.

4 Network Topology and Data Persistence

To demonstrate the decentralized nature of the application and ensure data consistency, we set up a private P2P network.

4.1 IPFS Swarm via Tailscale

Since default local IPFS nodes cannot easily discover each other over different NATs (Network Address Translations), we utilized **Tailscale** to create a secure overlay network.

1. **Setup:** Two computers were connected via Tailscale, assigning them virtual static IPs.

2. **Peering:** One machine acted as the "Server" (Host) and the other as the "Client."

3. **Swarm Connect:** We manually peered the IPFS nodes using the specific IPFS IDs and Tailscale IPs.

4.2 Data Persistence Results

This setup allowed us to verify that content uploaded by the Host was instantly discoverable and retrievable by the Client node. By pinning the content on both nodes, we achieved data redundancy, ensuring that even if one node goes offline, the data remains persistent on the network.

5 Conclusion

This project successfully demonstrates a functional prototype of a Web 3.0 social media platform. By integrating Ethereum Smart Contracts for logic and IPFS for storage, we achieved censorship resistance and data sovereignty. The implementation of SSI, Token-Gated encryption, and a zero-fee tipping economy offers a viable alternative to the exploitative models of Web 2.0. The successful networking test via Tailscale proves the system's capability to maintain data consistency across distributed nodes.