Reliable, Fair and Decentralized Marketplace for Content Sharing Using Blockchain

Prabal Banerjee
Indian Statistical Institute
mail.prabal@gmail.com

Chander Govindarajan
IBM Research - India
chandergovind@in.ibm.com

Praveen Jayachandran
IBM Research - India
praveen.j@in.ibm.com

Sushmita Ruj
CSIRO Data61, Australia
sushmita.ruji@data61.csiro.au

Abstract—Content sharing platforms such as Youtube and Vimeo have promoted pay per view models for artists to monetize their content. Yet, artists remain at the mercy of centralized platforms that control content listing and advertisement, with little transparency and fairness in terms of number of views or revenue. On the other hand, consumers are distanced from the publishers and cannot authenticate originality of the content. In this paper, we develop a reliable and fair platform for content sharing without a central facilitator. The platform is built as a decentralized data storage layer to store and share content in a fault-tolerant manner, where the peers also participate in a blockchain network. The blockchain is used to manage content listings and as an auditable and fair marketplace transaction processor that automatically pays out the content creators and the storage facilitators using smart contracts. We demonstrate the system with the blockchain layer built on Hyperledger Fabric and the data layer built on Tahoe-LAFS, and show that our design is practical and scalable with low overheads.

I. INTRODUCTION

Video and music on demand has emerged as a common delivery medium and business model that has spurred the media and entertainment market to grow consistently to tens of billions of dollars today. Two business models have been extremely popular and successful. First, a subscription model adopted by Netflix, Hulu and others, collects a fixed periodic subscription fee from consumers providing them unlimited viewing access to their repository of content. This provides the central facilitator disproportionate negotiating power and their contracts with content publishers are often lopsided, with the publishers getting no visibility into the viewership or popularity of their content and no share of revenue if the content is hugely popular [1]. A second business model based on advertisement revenue generated by platforms such as Youtube, makes the content typically free or available at a small pay-per-view charge, while making the bulk of their revenue through advertisements. Here again, content publishers only get a fraction of the revenue, typically only after they reach a certain level of popularity, with no transparency in how they are compensated [2].

In this paper, we aim to shift the balance more in favour of content publishers and provide a platform for them to showcase their content and be compensated fairly. We argue that this requires a third business model that is transactional in nature and disintermediates the central facilitator. A transactional pay-per-purchase model has worked well for music and books such as in Apple’s iTunes and Amazon’s Kindle, but there is little transparency and fairness to content publishers who need to completely trust the central facilitator on the number of copies sold. The key idea of this paper is to replace this central facilitator with a decentralized system of peers that collectively provide the same service of content listing and delivery. While similar decentralized marketplaces have been proposed by startups [3], [4], they do not provide adequate guarantees of reliability and fairness.

Our content sharing marketplace comprises of a decentralized data storage layer to store content in a fault-tolerant manner, where the peers additionally participate in a blockchain network. When a content publisher wishes to host their content on the platform, they first encrypt it and then apply erasure coding to break the content into smaller chunks. The chunks are stored on different servers in their storage layer, such that no single server has any content in its entirety. This ensures privacy of the content from the peers as well as tolerance to peer failures. A smart contract on the blockchain maintains a listing of all content and the hashes of the chunks held by different peers for each content. A consumer, when purchasing a content, downloads chunks from different peers and verifies that they match the hashes specified by the publisher. The consumer can then decode the original content and erasure coding ensures that the content is the original uploaded one. Our protocol ensures that a consumer cannot download the chunks from the peers and reconstruct the original content without making a payment. Upon receiving payment from the consumer, a smart contract on the blockchain ensures that the publisher as well as all peers serving the content are automatically paid in a fair and transparent manner. Further, we also support the ability to censor illegal content in a decentralized manner, preventing it from ever being downloaded.

We make the following key contributions in this paper:

- We develop a decentralized marketplace for content delivery that supports a transactional model for compensating publishers, by leveraging an innovative combination of blockchain, p2p data storage, and erasure coding.
- We guarantee (i) fairness for all parties involved despite maliciousness or collusion among peers, (ii) privacy of the content from peers involved in delivery, (iii) fault tolerance and availability of the content despite peer failures, and (iv) censorship of illegal, inappropriate, or expired content, all in a decentralized manner.
- We implement a prototype using Hyperledger Fabric [5] as the blockchain layer and the decentralized data storage
layer built using TahoeFS [6]. We study the performance and demonstrate that the system scales well with reasonable overheads.

II. SYSTEM MODEL AND PROBLEM STATEMENT

A. Background and Related Work

Digital Content Marketplaces: Digital content like ebooks, music and videos are all sold and purchased in central marketplaces that manage listing, discovery, secure storage, and purchase of content. More recently, with regulators clamping down on them, censorship of content is also being supported. However, the lack of physical connect between the seller and buyer that physical marketplaces offered, has caused fairness to be a concern. Content publishers do not have transparency on the number of copies sold and also get only a fraction of the proceeds from content sales [1], [7]. Regulators do not have auditability or the ability to track and censor illegal content [2].

Peer-to-peer Content Sharing: Existing P2P content sharing mechanisms such as BitTorrent [8] have evolved in a different context and do not provide a platform for a marketplace. Desired properties such as fairness, censorship and auditability can never be met. Issues like free riding [9] and blocked leecher [10] show that the system is not fair.

Decentralized Storage: Centralized cloud storage providers can be vulnerable to data thefts, leaks and downtime. With increased connectivity and availability of cheap storage, decentralized storage solutions such as IPFS [11] and Tahoe-LAFS [6] have emerged. They provide privacy and high availability as the system stores encrypted content on nodes forming a decentralized storage network. But, these solutions lack payment mechanisms to incentivize participants renting out storage. Also, the participants trust each other. Solutions like Storj [12], Filecoin [13] and others [14] use an incentive structure along with decentralized storage. Further, they support storage accountability by asking each node to produce proofs of storage, but do not support a marketplace with fairness to sellers and buyers. In our work, we leverage decentralized storage literature to design the storage and content delivery layer between a seller and a buyer.

Decentralized Marketplaces: Early work like Farsite [15] aside, recently some startups such as LBRY [4] and OpenBazaar [3] attempt to build decentralized marketplaces for digital goods. These systems do not guarantee content delivery after payment and do not address the issue of fairness. There are provisions such as multisig transactions in Bitcoin, where a trusted third party(TTP) can facilitate the payment but it is often hard to find a mutually trusted mediator. Once hosted, privacy of the content is also not ensured. Another major issue in such systems is that they are explicitly built to be censorship-resistant and not auditable, similar to Bitcoin. Governments either block indexing and tracking websites or ask the marketplace to pull down a content. But these techniques do not ensure that the content is inaccessible because of either mirror tracking websites, or only search results are blocked at the application layer on the client side and is not enforced in a decentralized manner. Clients can still easily find and download the illegal content. Similar solutions are presented in [16], [17], but they lack fairness and censorship properties provided in our design.

B. System Model

We have the following entities and roles in the system:

- **Publisher** ($P$) offers content on the system for purchase, but may not be able to host the content herself. She expects her clients to be able to discover her content and purchase them. To this end, she is willing to employ servers in the system who can store and serve her content at some cost.
- **Facilitator** ($S$) facilitates the marketplace by listing, storing and serving content. A collection of facilitators form the decentralized marketplace and they are compensated for their services. The system allows for independent server providers competing on costs and quality of service.
- **Client** ($C$) wants to browse, purchase and get content.
- **Auditor** is an external regulator who may impose restrictions on sale of particular illegal content and mark such content as not downloadable.

We assume a model where the publisher selects multiple facilitators for storing its content based on quality of service and cost. This avoids centralization and ensures fault tolerance. Given the above parties, several questions arise with regards to who pays the facilitators, the time of payment and the payment scheme. While many variants are possible, we discuss two broad variants:

1. Offline: The publisher and facilitators may have negotiations or conduct auctions with or without using blockchain for this process. The agreement could involve paying the facilitators a flat fee periodically or could be based on the number of downloads of the content. This process happens outside of the process of a buyer purchasing a content from the marketplace.
2. Online: When a buyer makes a payment to the marketplace, a smart contract executing on blockchain automatically pays the publisher as well as the facilitators involved in serving that content, based on payment terms previously agreed upon. The process of payment to the facilitators happens immediately with the buyer making the purchase.

Our proposed solution works with both these model variants. In both cases, the publisher decides the facilitators and then uploads to each of them. We do not explicitly discuss this process as there are well known methods leveraging auctions, negotiations and contracts. We do address the challenge of paying the facilitators in the online model. All content transfers happen point-to-point not involving the marketplace. However, operations such as payments are necessarily made via the decentralized marketplace.

We make the following assumptions in our model. We assume that each piece of content hosted on the platform is unique, identifiable by its hash. We also assume that while publishers, clients and facilitators may act maliciously and may collude, not more than a certain fraction of facilitators are simultaneously malicious for each content. Note that since the publisher chooses the facilitators, the client
colluding with one or more facilitators is a scenario that is less likely, although we still handle that scenario. We further assume that the auditor is non-malicious. We recognize centralized facilitators offer services like piracy prevention, advertisement and use of content delivery networks (CDN) which we do not directly address in our work. That said, our solution can easily be complemented by existing works like Digital Rights Management (DRM) solutions to combat piracy and supporting advertisements on blockchain [18].

C. Problem Statement

The problem we address in this paper is to ensure fair, reliable, privacy-preserving content delivery from a publisher to a client in a decentralized manner. Let \( \Pi \) be a protocol executed by a set of independent facilitators \( S \), with each facilitator \( S_i \in S \) having a file share \( F_i \) and a client \( C \) downloading a file \( F \) and paying price \( p \).

**Fair Exchange**: Protocol \( \Pi \) is fair if the following hold:
1) **Client Fairness**: If \( C \) pays \( p \) according to \( \Pi \), it is assured of getting the file \( F \) after termination of \( \Pi \).
2) **Facilitator Fairness**: For all \( i \in |S| \), if \( S_i \) inputs \( F_i \) according to \( \Pi \), it is assured of payment.
3) **Publisher Fairness**: If \( C \) pays \( p \) according to \( \Pi \) to purchase a file \( F \) published by \( P \), then \( P \) is assured of getting their share of the payment.

**Privacy**: Protocol \( \Pi \) preserves privacy of the content if executing \( \Pi \) does not reveal the content to any facilitator. The content should be revealed only to a client who has purchased it by making payment \( p \). A point to note here is that without the privacy property, fairness guarantee is impossible as any facilitator gains access to the content without payment, which is not fair to the content publisher.

**Censorship**: An auditor can place restrictions as follows:
1) Disallow any purchase of content deemed illegal
2) Restrict who can purchase certain content

**Audit trail**: For all transactions, the system is required to have a non-repudiable record of the buyer, publisher, content-id, and time for review by an auditor.

III. Solution

A. System Architecture

In a content sharing marketplace with no central authority, guaranteeing properties like fairness and privacy is a challenging problem. As two-party fair exchange is known to be impossible without a trusted third party[19], there is a need for mediation in trade. A decentralized blockchain network can replace the trusted intermediary, enforce fair trading rules and also manage payment. It naturally delivers transparency with its immutable replicated transaction log.

However, storing the digital content directly on the blockchain network would be impractical and not scalable. Hence, we need a storage layer comprising of storage servers along with the blockchain layer to complete the hosting of the marketplace. Having only a decentralized storage layer for mediation is not sufficient as these systems are meant to be cooperative and trusted, i.e., the storage servers do not compete with each other to store and serve files, and cannot guarantee immutability and fair payment. This is different from a blockchain setting where mutual distrust between the peers is tolerated. These two layers can be operated independently by different entities as long as each storage server can query some peer in the blockchain layer. For ease of exposition, in the sections that follow, a facilitator \( S_i \) refers to a storage server with access to a blockchain peer (possibly both operated by the same facilitator).

Publisher \( P \) and client \( C \) only communicate with the facilitators. The publisher \( P \) needs to know some subset of facilitators available in the system along with access to a blockchain node for performing transactions. The client \( C \) should have access to a blockchain node for making queries and invokes to the ledger.

The storage layer, if supplied with unencrypted content, would violate privacy. Also, a decentralized network may be collectively honest, but any individual facilitator may be malicious and cannot be trusted. Therefore, the publisher \( P \) first erasure codes a file into chunks, encrypts each chunk, and sends each encrypted chunk to a different facilitator in the network, such that privacy is preserved and the system guarantees fault tolerance. We explain the detailed protocol and the algorithms in the sections that follow.

Our use of the blockchain network is quite generic and most platforms with support for smart contracts should work. However, a permissioned blockchain with stronger notions of identity and support for access control, is better suited for enforcing properties like censorship, for eg. roles such as regulator or auditor with specific rights to flag content as illegal or invalid.

B. Notation and Preliminaries

We use ledger \( L \) to signify the blockchain layer hosting the smart contracts. We additionally assume that all parties register with \( L \), and the public key of each party is available with \( L \) before start of the protocol. We associate a Uniform Resource Identifier (URI) with every file \( F \) which unambiguously refers to that particular file. We use Erasure Coding to encode the file \( F \) into \( n \) chunks, such that any \( k \) out of \( n \) are sufficient to recover the data. We refer to this ratio \( k : n \) as the replication ratio.

C. Content Sharing Protocol

Figure 1 shows the entire protocol at a glance, which is divided into two phases. The content upload phase deals with a publisher distributing its content to the facilitators. In the content delivery phase, the client fetches the content from the facilitators and incentives are distributed.

1. Content Upload Phase: The Publisher \( P \) wants to monetize \( F \) and needs to supply \( F \) to its chosen set of facilitators. It generates \( URI_F \) which is used to uniquely identify \( F \). It breaks \( F \) into multiple erasure coded chunks \( F_1, \ldots, F_n \), such that any \( k \) out of \( n \) chunks are enough to reconstruct \( F \). \( P \) generates a set of keys for convergent encryption, with which it encrypts the file chunks \( F_i \) to obtain \( E_i \). It generates a hash list of these encrypted file
chunks so that any potential buyer can verify integrity of received data. Finally, it uploads the file chunks, one chunk to each facilitator, and then records the encryption key set, price, URI and the hash list to the ledger.

2. Content Delivery Phase: In this phase, \( C \) wants to purchase a particular file referred by \( URI_F \). It discovers this file through the content listing on the blockchain. First, it queries the ledger to fetch the price \( p \) of the file and generates a unique request identifier \( reqID \). \( C \) then pays amount \( p \) to the ledger. The ledger accepts this payment only if the requested content is not censored by an auditor. Upon successful payment, it gets the decryption key map and the list of file chunk hashes from the ledger. It first approaches any \( k \) facilitators, fetches file chunks and matches their hashes with the hash list. If some facilitators do not respond or respond incorrectly, \( C \) approaches additional facilitators until it gets \( k \) correct file chunks. Finally, it uses the key map to decrypt the file chunks and then recovers the original file.

\( S_j \) on receiving a file chunk first verifies its integrity using the key map submitted on the ledger by \( P \). That is, decrypting the received content using the hash of the unencrypted file chunk as the key, and then taking its hash, must reveal the same hash. If the check fails, \( S_j \) raises a complaint. If more than \( n - k \) complaints are raised for a file, the file is not made available for clients to download. This ensures that a publisher cannot provide malicious content to facilitators. A key step during content delivery is that each honest facilitator checks whether payment has been made on the blockchain before serving content to a client. If the payment is not done, honest \( S_j \) should deny service.

Payment: In section II-C, we had defined fairness from a client and a facilitators perspective. In our proposed design, the client fetches parts of content from the facilitators after payment. This interaction between \( C \) and \( S \) contains no intermediary or verifiable log. Hence, ascertaining whether a facilitator has actually served content or not is hard as a malicious facilitator would always claim to have served to maximize revenue. Similarly, a malicious client would also claim to have not received content, if she only has to pay the ones whom she claims to have got service from. To counter this, our protocol enforces the client to pay a fixed amount before downloading content and the blockchain transfers the payment equally to all the facilitators serving chunks of URI irrespective of whether they serve a particular \( reqID \). The pseudocodes of our algorithm and analysis of the impact of facilitator incentivization on issues like availability, free-riding, failures and latency is present in [20].

D. Design Analysis

In this section we argue that our proposed design satisfies the goals outlined in section II-C. Our presumed adversarial model allows the adversary to control all the malicious parties together. However, we note that \( P \) chooses the facilitators. Hence, a collusion among the client \( C \) and facilitator \( S_i \) is unlikely. Also, \( C \) and \( P \) has no direct interaction in our protocol and hence their collusion case reduces to individually malicious cases.

**FE1. Client Fairness**: Our protocol II ensures that an honest client who has paid \( p \) to \( L \) should get access to \( F \). In particular, it should get access to \( k \) valid chunks and their decryption keys. Under the assumption that the blockchain is tamper-proof and it can tolerate up to \( b \) out of \( n \) malicious nodes, then as long as \( k < (n - b) \), there are at least \( k \) honest facilitators in the system. These honest facilitators would follow the protocol and serve chunks to \( C \). This would allow \( C \) to decrypt and reconstruct the file, as guaranteed by erasure coding. The smart contract execution is also assumed to be tamper-resistant and hence malicious parties cannot affect the key release by \( L \) upon successful payment. Thus, client fairness is guaranteed.

**FE2. Facilitator Fairness**: In our protocol II, all facilitators \( S_i \) are paid according to the pay-off function \( p_o \), irrespective of whether they have served. Assuming the smart contract execution is tamper-resistant, payment to all facilitators is ensured by \( L \), regardless of whether they served \( C \). Thus, our protocol II guarantees facilitator fairness.

**FE3. Publisher Fairness**: For every valid transaction, the payment is made to \( L \). As the smart contract pays directly to \( P \), II guarantees publisher fairness, unless smart contract execution is tampered with.

**Privacy**: According to II, no facilitator holds more than one chunk. An adversary needs access to at least \( k \) chunks to obtain file \( F \). Under the assumption that \( k > b \), the adversary will not be able to reconstruct the file, even though it may control \( b \) facilitators and can obtain the decryption keys from the ledger. Hence, no facilitator can gain access to the whole file and only a client who has paid will be allowed to download \( F \).

**Censorship**: In our proposed protocol II, \( L \) maintains a list of censored content. Only Auditors have write access to this list. When \( C \) invokes \textit{Payment()} for a particular \( URI \), \( L \) checks whether that \( URI \) is on the list. It only executes the request if the content is not censored. As long as the blockchain layer is tamper-resistant, payment will not be made for a \( URI \) present on the censor list. Every honest
facilitator $s_i$ checks with $L$ before serving content, and hence as long as $k > b$, the adversary will not be able to access a censored content. Hence, $L$ supports censorship.

As discussed above, the ideal bound for values of $k$ is $b < k < (n - b)$, where the lower bound ensures privacy & censorship and the upper bound ensures fairness. This might sound like a tricky bound for PoW based systems like Bitcoin [21] where $b < n/2$, but recent works [22] show that the practical bound is much lower than that. Other consensus like PoS and BFT already have bounds $< n/2$. For example, PBFT [23] can tolerate $< n/3$ faulty nodes among $n$ nodes. Hence for PBFT, the bounds for $k$ would be, $n/3 < k < 2n/3$.

**Audit Trail:** All the content upload, requests and payments go through the smart contract, leaving an append-only trail of transactions. Any auditor can just query the blocks and gain full information on the trail of transactions.

**IV. Implementation**

For our prototype, we chose Hyperledger Fabric [5] as the blockchain layer and Tahoe-LAFS [6] as the storage layer. A facilitator may run a Tahoe server and/or a Fabric peer. Hyperledger Fabric is a permissioned blockchain platform with support for smart contracts that scales well in terms of transaction throughput and commit latency. Each facilitator operates a Fabric peer and runs an instance of our marketplace smart contract. This smart contract maintains the content listing and provides the interface to make and verify payments. A subset of the facilitators, auditors, authorities and independent third parties would form the ordering service with access control rules setup to allow only auditors to edit blacklists.

Tahoe-LAFS [6] (Least Authority File Store) is a cooperative distributed file system platform that provides security and fault tolerance guarantees. The erasure coded and encrypted file chunks are stored on a set of nodes called Tahoe Servers each of which may crash independently. Tahoe provides a reliable data store capable of tolerating such faults based on the parameters of the file encoding (the replication ratio).

Publishers and buyers are implemented as scripts built over the Tahoe client with an additional 100 lines of change in the Tahoe-LAFS Python codebase. All components were run as Docker containers orchestrated using Swarm.

**V. Evaluation**

The primary metric for any content delivery marketplace is the latency experienced by a client for content download which we define as the time taken from the point the client initiates a purchase until it receives the entire content. We measure the overhead in the delivery latency of our solution compared to a baseline decentralized file storage, to demonstrate that the solution is practical. We study this by varying several parameters including file size, replication ratio, load, number of facilitators, inter-node latency, and the number of simultaneous failed nodes in the system.

We evaluate our prototype on 7 VMs in a single data center all running Linux (Ubuntu 16.04) configured with 16 vCPUs, 32 GB RAM and 100 GB HDD storage. Unless specified otherwise, we use the following default values for our experiments: file size of 10 MB, a replication ratio of 4:6, and 6 facilitators. In each experiment, we vary a different parameter keeping other parameters constant, to understand its impact on performance. In the default cases, each of the facilitators and the client are run on distinct VMs. In experiments with more participants, multiple clients and facilitators are co-located on these VMs, in a load-balanced manner. Each experiment is run 5 times and the averaged observations are reported.

**Varying file size:** We first study the impact of content file sizes on client latency. Figure 2 shows the latency impact for both upload and download of files in the presence and absence of blockchain, for file sizes ranging from 100 KB to 100MB. As expected, the latency increases sub-linearly with file size. For both upload and download, the overhead of using blockchain is within 10% of the baseline. We note that the upload and download with blockchain consist of the entire Content Upload and Delivery Phases respectively, as illustrated in Figure 1.

**Varying replication ratio:** An important parameter in our solution is the replication ratio of erasure coding. It represents the level of decentralization, fault tolerance and availability. We present results for $n = 6$ and $k$ varying from 2 to 6 in Figure 2. The upload times remain largely unaffected as the only changing factor is the file encoding time because of changing replication ratio. The download times (both with and without blockchain) increase with $k$, as more chunks have to be downloaded to reconstruct the original file. For all replication ratios, the overhead of using blockchain is nearly constant for both upload as well as download. Additionally, we performed experiments where $n$ was varied keeping $k$ fixed (not shown due to space constraints). The upload time increased with increasing $n$ as more chunks needed to be uploaded, but the download times were nearly constant.

**Varying number of facilitators:** We next show that our system scales horizontally by increasing the number of facilitators and evenly distributing the load among them. As the replication ratio remains fixed with increasing facilitators, a constant number of servers need to be contacted for file upload and serving of the chunks, causing file sharing latency to be unaffected. However, the latency of each blockchain transaction likely increases with the number of peers. Figure 2 shows that there is no impact on latency for variation in the number of facilitators from 6 to 24 in steps of 6, indicating that the bottleneck is not with the latency of blockchain transactions, but rather with the file sharing.

**Varying load:** Next, we study how latency observed by a client varies with overall system load, the results of which are presented in Figure 3. Here, the x-axis shows the number of clients operating simultaneously, for both upload and download scenarios (when we measure latency
In this paper, we presented a decentralized and fair marketplace for content sharing that supports a pay-per-purchase model of compensating content producers. By leveraging a novel combination of blockchain, peer-to-peer storage and erasure coding, our system guarantees fairness to all participants despite presence of maliciousness or collusion, privacy of content from peers involved in delivering content, configurable fault tolerance and availability, and support for censorship of illegal content, all without reliance on a central facilitator. We evaluate our system and show that it scales well with low overhead. Designing a model to address streaming content and a reputation based incentive structure is part of future work.

VI. CONCLUSION

In this paper, we presented a decentralized and fair marketplace for content sharing that supports a pay-per-purchase model of compensating content producers. By leveraging a novel combination of blockchain, peer-to-peer storage and erasure coding, our system guarantees fairness to all participants despite presence of maliciousness or collusion, privacy of content from peers involved in delivering content, configurable fault tolerance and availability, and support for censorship of illegal content, all without reliance on a central facilitator. We evaluate our system and show that it scales well with low overhead. Designing a model to address streaming content and a reputation based incentive structure is part of future work.

REFERENCES

[1] Netflix Secrecy Over Its Ratings is an Unfair Advantage, Say Rivals. [Online]. Available: https://bit.ly/3mPtq3X.
[2] Criticism of Google#Youtube. [Online]. Available: https://en.wikipedia.org/wiki/Criticism_of_Google#YouTube.
[3] OpenBazaar. [Online]. Available: https://openbazaar.org.
[4] LBRY. [Online]. Available: https://lbry.com.
[5] E. Androulaki, A. Barger, V. Bormikov, C. Cachin, K. Christidis, et al., “Hyperledger fabric: A distributed operating system for permissioned blockchains,“ in Proc. ACM Eurosys, 2018.
[6] Z. Wilcox-O’Hearn and B. Warner, “Tahoe: The least-authority filesystem,” in Proc. of ACM Workshop on Storage Security and Survivability, 2008, pp. 21–26.
[7] BBC, Spotify settles $1.6bn lawsuit over songwriters’ rights, Dec. 2018. [Online]. Available: https://bbc.in/3kwYfrA.
[8] B. Cohen, Incentives Build Robustness in BitTorrent, May 2003. [Online]. Available: http://bittorrent.org/bittorrentecon.pdf.
[9] J. Wang, R. Shen, C. Ullrich, H. Luo, and C. Niu, “Resisting free-riding behavior in BitTorrent,” Future Generation Computer Systems, vol. 26, no. 8, pp. 1285–1299, Oct. 2010.
[10] A. Ramachandran, A. das Sarma, and N. Feamster, “Bitstore: An incentive-compatible solution for blocked downloads in bittorrent,” in Proc. Joint Workshop on The Economics of Networked Systems and Incentive-Based Computing (NetEcon), 2007.
[11] J. Benet. (2014). “IPFS - Content-Addressed, Versioned, P2P File System,” arXiv: 1407.3561v1 [cs.NI].
[12] S. Wilkinson, T. Boshevski, J. Brandoff, and V. Buterin, “Storj a peer-to-peer cloud storage network,” 2014.
[13] P. Labs, Filecoin. [Online]. Available: https://filecoin.io.
[14] V. He, H. Li, X. Cheng, Y. Liu, C. Yang, and L. Sun, “A blockchain based truthful incentive mechanism for distributed p2p applications,” IEEE Access, vol. 6, pp. 27 324–27 335, 2018.
[15] A. Adya, R. P. Wattenhofer, W. J. Bolosky, M. Castro, et al., “Farsite,” ACM SIGOPS Operating Systems Review, vol. 36, no. 5, Dec. 2002.
[16] M. Klens, J. Eberhardt, S. Tai, S. Härteiln, S. Buchholz, and A. Tidjani, “Trustless intermediation in blockchain-based decentralized service marketplaces,” LNCS, pp. 731–739, 2017.
[17] O. R. Kabi and V. N. Franqueira, “Blockchain-based distributed marketplace,” in International Conference on Business Information Systems, Springer, 2018, pp. 197–210.
[18] B. Wise, Blockchain for advertising: The new black for media buying. [Online]. Available: https://ibm.co/2ZRXBNM.
[19] M. O. Rabin, How to exchange secrets with oblivious transfer, Cryptology ePrint Archive, Report 2005/187, 1981.
[20] P. Banerjee, C. Govindarajan, P. Jayachandran, and S. Ruj. (2020). “Reliable, Fair and Decentralized Marketplace for Content Sharing Using Blockchain.” arXiv: 2009.11033 [cs.CR].
[21] S. Nakamoto, “Bitcoin: A peer-to-peer electronic cash system,” 2008.
[22] I. Eyal and E. G. Sirer, “Majority is not enough: Bitcoin mining is vulnerable,” Commun. ACM, vol. 61, pp. 95–102, 2013.
[23] M. Castro and B. Liskov, “Practical byzantine fault tolerance,” in Proc. USENIX OSDI, 1999, pp. 173–186.