LBRY: A Blockchain-Based Decentralized Digital Content Marketplace

Jun Li*†, Alex Grintsvayg*, Jeremy Kauffman*, Charles Fleming‡
*LBRY Inc., USA †University of Oregon, USA ‡University of Mississippi, USA
Email: lijun@uoregon.edu; grin@lbry.com; jeremy@lbry.com; fleming@olemiss.edu

***Abstract*—Despite the critical need of publishing and consuming content online, a centralized content platform such as Amazon or Youtube may not always have their policy and practice aligned with the interest of their users and could be rent-seeking, censorious, and frequently exploitative, whereas a peer-to-peer solution such as BitTorrent may suffer from issues of content discovery, legitimacy, monetization, verifiable publisher identity, and poor user experience.**

In this paper, to improve significantly over both options, we propose a new approach called LBRY that enables a decentralized online content marketplace. In particular, it uses a blockchain to build a decentralized content platform controlled by the community, and allows its users to publish, host, find, access, download, and pay for content with ease. LBRY introduces a new naming scheme that gives users the full control of the names of their content, and uses a blockchain to not only support a digital currency (LBC) and transparent decentralized ledger, but also allow every user to access a synchronized name space and a global index of content metadata, thus supporting a new paradigm of digital content distribution. We detail how LBRY works in this paper, including how it designs its data structures for content, content metadata, and a novel content naming space; how it uses the blockchain to manage and synchronize the name space and implement an index for content metadata in order to support content sharing and purchase; and how it handles several issues in running LBRY.

***Index Terms*—decentralized content marketplace, blockchain, content naming, digital content, distributed hash table (DHT)**

I. Introduction

People have a fundamental and ever-growing need to publish, host, find, download, and pay for content—books, movies, music, or anything else that can be represented as a stream of bits. Usually, people have to choose between a centralized host or platform such as Amazon or YouTube, or a peer-to-peer protocol such as BitTorrent [1].

Centralized platforms suffer from several problems because their incentives are not aligned with the incentives of their users. They often engage in rent-seeking behavior, including extracting a large percentage of content creator profits. They may enforce opaque and arbitrary rules on content creators, and can change those rules without warning or community input. They may choose to censor content at the behest of repressive regimes around the world, or in a profit-maximizing consession to social pressures.

BitTorrent does not have these faults, but it has problems of its own. It is only useful if one already knows the info hash of the content they seek, and there is no way to discover these hashes within the protocol. Even using an external search engine does not provide a comprehensive list of what is available on the network. There are no incentives for users to seed content, and BitTorrent largely works because users earn status through private communities, are nice, or simply fail to understand what their client is doing. There is no way to earn revenue from published content, and no identity mechanism to build a fan following. Finally, a lot of content on BitTorrent infringes on copyright, which taints the protocol's public perception and overshadows the many positives it has.

LBRY offers a significant improvement over both options. It allows its clients to publish, host, find, access, download, and pay for content in a global, decentralized marketplace. It uses a public blockchain to build a decentralized content platform controlled by the community to replace the good parts of a centralized host (a common interface to store data, find interesting content, build a brand, pay for content, and get rewarded for contributing), while removing the downsides (opaque and arbitrary rules, rent extraction, censorship). The blockchain simultaneously solves several problems:

- It stores metadata about every piece of content on the network, creating a complete index of all published content. This index facilitates content discovery.
- The index is public. Anyone can use it to build a better search engine or user interface, reducing platform risk and making rent extraction more difficult.
- A public index also makes copyright infringement more difficult to hide.
- Proof of work is required to update the blockchain, so any censorship or rule changes must have a strong community support.
- The blockchain provides an in-protocol payment mechanism. Creators, developers, and service providers can be paid for their efforts.
- Verifiable identity becomes as simple as publishing a public key to the chain.

LBRY has been in public use since June 2016. It has been designed and tuned based on extensive real-world feedback. Approximately 2 million pieces of digital content have been published on LBRY, and hundreds of thousands of users access millions of files each month, downloading and uploading terabytes of data. Graphical browsers and wallets are available for all major operating systems and can be downloaded at https://lbry.com/get [2].

978-1-7281-6978-1/20/$31.00 ©2020 IEEE
DOI 10.1109/DAPPS49028.2020.00005

II. Related Work

Most digital content on the Internet today is distributed through a combination of a web site which handles payment, authentication, and authorization, as well as metadata for the content, and a *Content Delivery Network (CDN)* which handles content delivery to the user [3]. Traditional CDNs use a request routing system to forward user requests to the most suitable edge server [4]. As an alternative to CDNs, peer-to-peer (P2P) systems distribute content via peers that act as both clients and servers and contribute storage and bandwidth to the network [5]. P2P systems can be broken into roughly two groups: partially centralized, where special nodes control or manage the content swarm, and completely decentralized, where all nodes are identical. A popular design for partially centralized P2P systems is to utilize *tracker nodes* which store metadata such as file hashes and available peers [6].

Blockchain technology came to prominence as the underlying technology of Bitcoin [7], and has since been generalized to a distributed ledger that records transactions in *blocks* that are cryptographically linked to previous blocks. This design has been used to create several systems related to content management. One of the first of these was Bright, a digital rights management system built on the Bitcoin blockchain [8], which was limited to storing rights information as part of the blockchain. This was extended in [9] to include storage of image region hashes on the blockchain and embedding of transaction ID numbers as watermarks in images, allowing users to verify both ownership of an image and that it was unmodified. In [10] the authors describe a secure P2P-based file storage system where encrypted files are stored on peer nodes and metadata, including decryption keys, are stored on a blockchain. The Ushare system proposes a similar system, with user shared data stored encrypted on a distributed hash table based P2P system and metadata stored on a blockchain, allowing the user to control who has access to his shared media. More focused on licensing of digital media, the Monegraph system provides a digital marketplace for buying and selling digital art, but does not include the ability to store or download these items [11].

More general systems have also been created leveraging the idea of storing metadata on a blockchain. Metadisk is one example of this, built on top of the Storj P2P storage system [12] [13]. Another is BigchainDB, a scalable database designed for storing big data utilizing blockchain technologies. As part of its central design, BigchainDB includes ownership information of all data it stores [14].

III. Overview

LBRY's goal is to create a digital content market for its users without any centralized authority or point of control. It is based on the LBRY network composed of LBRY nodes, where every LBRY user can run a LBRY client software with a familiar user interface on a LBRY node to interact with the LBRY network, including publishing any content, such as a video or a spreadsheet or any file, searching, accessing, and downloading content published by other LBRY users for free or with a payment, or simply passively participating in the network to automatically earn rewards in exchange for contributing bandwidth, disk space, or processing power to the overall network. Figure 1 is a screen shot of a LBRY browser in use.

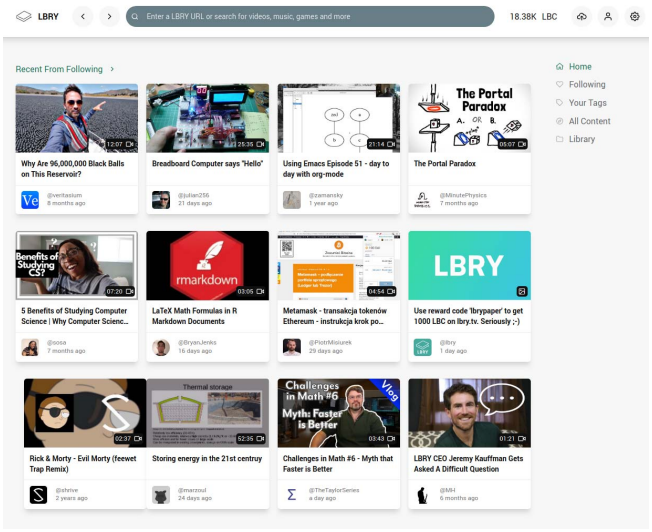


Fig. 1: **A screen shot of the LBRY desktop app.**

Here is a sample use of LBRY: Ernest releasing a film on LBRY that is later purchased and viewed by Hillary.

1) Ernest wants to release his comedy-horror film, Ernie Runs For President.
2) The content is encrypted and split into many pieces. These pieces are made available on the LBRY network.
3) Ernest claims the name `ernieruns`. The URL `lbry://ernieruns` points to that claim.
4) When Ernest claims the name, he also submits other metadata of the content, such as a description, thumbnail, pricing information, and content hash.
5) Hillary, a user, opens her browser, searches the LBRY network, and decides she wants to watch the film at `lbry://ernieruns`.
6) Hillary issues a payment to Ernest for the decryption key, allowing her to watch the film.
7) Hillary's LBRY client uses the content hash in the metadata to collect the pieces from the hosts. It reassemble them, using the key to decrypt the pieces (if necessary). This is transparent to Hillary, and the film streams a few seconds after purchase.

From a user's perspective, the interaction is extremely similar to those that happen on hundreds of different web sites (e.g., YouTube, Amazon and Netflix). The key difference is that this one happens via a network that is completely decentralized. The data and technology that makes the entire interaction possible is not reliant on nor controlled by any single entity.

The LBRY system incorporates three main data structures to meet its goal:

- **Distributed Hash Table (DHT) for content data.** LBRY uses DHT to connects LBRY nodes to each other and store and access content data in LBRY nodes. Any client that hosts content uses the DHT to announce that it has a particular content, using the hash of the content. Any client looking to download a particular content, which is referred by its hash, uses the DHT to find whom to download it from.
- **Blockchain for a consistent, verifiable storage for content metadata, publisher identity, and payment records.** LBRY uses a public proof-of-work blockchain to store content metadata and index content available on the network. The blockchain also records content purchases and maintains a system of pseudonymous identity for publishers.
- **Claimtrie for mapping names to metadata.** LBRY has a novel naming system to create human-friendly identifiers for content and aid in discovery, while avoiding name squatting (described in more detail in Section V). It uses the Claimtrie, a modified Merkle tree, to map the name of a claim to the metadata for that claim. Clients can publish "claim operations" to the blockchain to manipulate the Claimtrie, thereby adding, updating, and removing content in the global index. The root hash of the Claimtrie is entered into the block header of each block on the chain. This allows clients to verifiably resolve claims even without a full copy of the blockchain data (especially useful for mobile clients and clients with low-bandwidth connections or limited disk space).

To understand how these components work together, we consider two use cases. The first case is a publisher adding content to the LBRY network (e.g., Ernest in the above sample use of LBRY). To accomplish this, the publisher selects a name for the content and inputs other metadata (e.g. title, description, price). The publisher may choose to associate this content with a pseudonymous on-chain identity. The publisher's client encodes this metadata into a blockchain transaction to insert it into the Claimtrie, and broadcasts the transaction to other nodes. Concurrently, the client announces the availability of the content via the DHT. When the block containing this claim is added to the blockchain, each node updates its copy of the Claimtrie using the claim operations from the block (including inserting the publisher's claim). The second case is a customer purchasing content corresponding to a particular LBRY URL (e.g., Hillary in the above sample case of LBRY). In this case, the client software will first query a blockchain node for the metadata associated with the URL. The node will look up the URL in its Claimtrie and return the metadata to the client. The client can then use this data to purchase the right to access the content, which it then downloads from the LBRY network via DHT.

IV. Content

A. Stream for File Encoding and Decoding

Every file in LBRY must be encoded into a **stream** before it can be published and shared. Specifically, every stream begins with a manifest blob, followed by one or more content blobs that hold the actual content of the stream. Here, we break content into blobs to facilitate partial and distributed hosting. Moreover, we encrypt blobs to reduce infringing content access (i.e., accessing paid content without paying for it) and commodify provision of data hosting (hosts may choose to not know what content they host). Content blobs are encrypted using AES-256 in CBC mode and PKCS7 padding, with each encrypted blob 2MiB maximum.

The manifest blob contains information necessary to find the content blobs and decode them into the original file. In particular, encoded using canonical JSON encoding [15], the manifest blob has the following fields:

- *blobs*: An ordered description of content blobs in the stream. Each item in the list has the **blob hash** for a content blob (which in the current version of LBRY is a SHA-384 hash), the hex-encoded initialization vector used to create the blob, and the length of the encrypted blob.
- *filename*: The hex-encoded name of the original file.
- *key*: The hex-encoded stream key used to decrypt the blobs in the stream. This field is optional. The stream key may instead be stored by a third party and made available to a client if the client is authorized to receive the key.
- *version*: Currently 1. It is intended to signal structure changes in future versions of this protocol.

The hash of the manifest blob of a stream is also called the **stream hash** of the stream. It uniquely identifies the stream and can be used to locate and fetch the stream from the LBRY network.

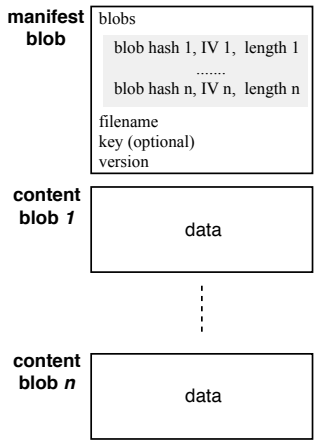


Fig. 2: **Stream format in LBRY.** Every file in LBRY is encoded into a stream.

Decoding a stream into a file is like encoding in reverse, with the added step of verifying that the expected blob hashes match the actual data. Specifically, it includes the following steps:

- Calculate the hash of the manifest blob and verify the value matches the stream hash provided.
- Parse the manifest blob.
- Calculate the hash of every content blob and verify the value matches its hash recorded in the manifest blob.
- Decrypt and remove the padding from each content blob using the stream key and IVs in the manifest blob.
- Concatenate the decrypted content blobs in order and finally retrieve the file, with the file name assigned according to the manifest blob.

B. Channel as Publisher Identity

In order to allow content to be associated with specific pseudonyms and identities, LBRY introduces a concept called **channel** as the unit of publisher identity, where every channel has a pair of public key and private key. This concept is a substantial improvement over BitTorrent, which has no concept of provable identity. The channel concept allows publishers to aggregate their content, maintain attribution, and build a following. Users can follow channels and be notified when new content is added. Channels also give publishers a human-memorable way of referring to their content (e.g., lbry://@Veritaseum)

C. Content Sharing via DHT

LBRY enables file storage and access without relying on centralized infrastructure, creating a marketplace for data that allows hosts to be paid for their services. In particular, the LBRY clients use a distributed hash table (or DHT) to build a peer-to-peer content network, i.e., the LBRY network, to publish, search, access, and download LBRY files. As every file is encoded into a stream with a manifest blob and one or more content blobs, LBRY uses DHT to store blobs at peer nodes in the network, announce blobs, track these blobs, and discover peers that host each blob. A DHT is a key-value store that is spread over multiple nodes in a network. Nodes may join or leave the network anytime, with no central coordination necessary. Nodes communicate with each other using a peer-to-peer protocol to advertise what data they have and what they are best positioned to store. LBRY's DHT implementation follows the Kademlia [16] specification fairly closely, with some modifications.

When a host has a file available for download, it must encode the file into a stream (Section IV-A), including generating a manifest blob that contains the hash of every content blob. Then, for every blob it wishes to share, it can announce via DHT and let other nodes on the LBRY network know the hash of the blob, in two steps: (1) The host uses the Kademlia protocol to look for nodes (i.e., peers) that are "closest" to the hash: the notion of closeness between a hash and a node is defined as the bit exclusive or (XOR) between the hash value and the ID of the node (which have same number of bits), as defined in Kademlia. (2) The host asks those peers to store the fact that the host in question has the blob available for download.

If a client wishes to download a stream, it must first obtain the manifest blob of the stream based on the stream hash of the stream (recall it is also the hash of the manifest blob), and then obtain every content blob of the stream based on the hash of the blob as indicated in the manifest blob. The procedure for obtaining a blob based on its hash is as follows: (1) Similar to the above first step of content announcement, the client queries the DHT to find nodes that are "closest" to the hash; (2) It queries those nodes to learn the hosts that have the blob and announced the hash of the blob; and (3) It contact those hosts to obtain the blob. Once the client has all the blobs of the stream, it then can decode the stream to obtain the original file (Section IV-A).

One requirement for content sharing to work is that there must be hosts online that have the content a client wants when the client wants it. To incentivize the continued hosting of data, LBRY enables hosts called **reflectors** to accept data uploads from the original hosts such that clients can also download data from reflectors. Using a reflector is optional, but most publishers will probably choose to use them in order to obviate the need for the publisher's server to be online and connectable, which can be especially useful for mobile clients or those behind a firewall.

V. Content Metadata

In supporting a digital content marketplace, LBRY allows a user to make a **claim** of every stream or channel she has such that other users can learn their metadata and make purchases. We describe how LBRY manages transactions of a content marketplace through a blockchain in Section VII. In this section, we focus on the claim data structure and describe how claims are defined and managed.

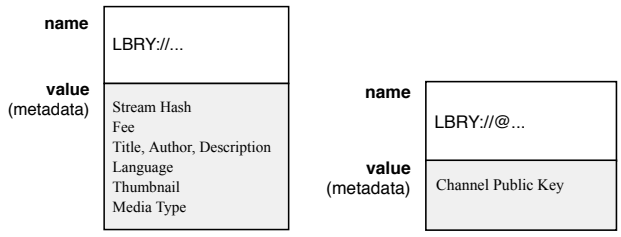
A. Claims: Data Structure for Stream and Channel Metadata

LBRY uses claims to store and manage the metadata of the content available on the network. Specifically, for every stream it uses a **stream claim** to store and manage the metadata of the stream. Also, for every channel it uses a **channel claim** to store and manage the metadata of the channel. Every claim has a **name** field and a **value** field. The name field is an UTF-8 string of up to 255 bytes to name the claim. If it is a channel claim name, the first character of the name is '@'. The name also leads to a **LBRY URL** with the format `lbry://<name>` that points to the claim. The value field is the metadata of a stream or a channel. The metadata of a channel is the channel's public key. The metadata of a stream, however, includes more fields, including:

Stream Hash: A unique identifier that is used to locate and fetch the content from the data network. As described in Section IV-A, the stream hash of a stream is the blob hash of the manifest blob of the stream.

Fee: Information on how to pay for the content. It includes the address that will receive the payment (the fee address), the amount to be paid, and the currency.

Title, Author, Description: Basic information about the stream.



(a) Stream Claim (b) Channel Claim

Fig. 3: **Claim data structure.**

Language: The ISO 639-1 two-letter code [17] for the language of the stream.

Thumbnail: A URL to be used to display an image associated with the content.

Media Type: The media type of the item as defined by the IANA [18].

Figure 3 shows the data structure of a stream claim and a channel claim.

B. Claimtrie: Data Structure for Storing Claims at Clients

LBRY stores the entire set of claims in a data structure called **Claimtrie**, a trie data structure [19]. Just as a trie is an ordered tree of nodes which supports the *Find* operation to returns the value for a key and the *Insert* operation to insert a key and a value into the trie, both in O(n) time, where n is the length of the key, the Claimtrie uses the names of claims as the keys and the claims themselves as the values to order and organize the claims. Specifically, names are stored as the paths from the root node to the leaf nodes, and claims are stored as leaf nodes in the tree.

Every client can have a copy of the Claimtrie, synchronized through the LBRY blockchain as we will show in Section VII.

To avoid confusion due to Unicode equivalence or casing, names in the Claimtrie are normalized when performing any comparisons or determining the Claimtrie path to the leaf node for a name. When names are being compared, they are first converted using Unicode Normalization Form D [20], then lowercased using the en_US locale. This means names are effectively case-insensitive.

Multiple claims can exist and compete for the same name. They are all stored in the leaf node for that name. Sec. VII-E describes the algorithm that determines the order of claims in a leaf node.

VI. Content Naming

A. Design of Naming

LBRY designs a naming system such that a stream or a channel can have a human readable name that is expressed as a LBRY **URL** and can be mapped to a unique, permanent ID representing the stream or the channel. Particularly, it departs from most existing name allocation designs, which mostly are first-come, first-serve with a fixed price, such as the domain name system (DNS). This leads to several bad outcomes:

- *Compulsory control.* Traditional naming schemes usually exercise compulsory control on names. Name holders (except the top-level provider) do not have true ownership of their domain. While they can allow for an authority to use a process to reassign names, such a process is often arbitrary, changes over time, and could even be a censorship point.
- *Rigid transaction cost.* A traditional domain name scheme usually involves significant transaction costs. Moreover, as it usually does not allow a price to float freely, its price control is inefficient. If a price is set too low, there is speculation and rent-seeking. If the price is too high, users are excluded from a good that would otherwise be beneficial for them to purchase.
- *Speculation and extortion.* Entrepreneurs are incentivized to register common names even if they do not intend to use them, in hopes of selling them to the proper owner in the future for an exorbitant price. Speculation also harms the user experience, who will see the vast majority of names sitting unused.

The naming system embraces the following design goals:

- *Discretionary control.* Every user can name their own content at their own discretion; no approval from any authority is needed. In fact, it allows creators to acquire a URL as the name of a stream or a channel, which can be as short and memorable as possible.
- *No transaction fee.* Once a name is created, the creator can own it permanently and forever, without ongoing fees.
- *No speculation and extortion.* It prevents squatters from extorting creators. While a single word with no other extension or modifier can be mapped directly to a piece of content, multiple pieces of content can be located at the same keyword.

B. Name Resolution

A critical function of LBRY is for a client to resolve the name of a stream or a channel, a LBRY URL, to the claim of the stream or the channel, respectively. The resolution is through the Claimtrie, a trie data structure described in Section V-B that LBRY employs to store the entire set of claims against their names. One challenge here is that because LBRY allows every use to name their own content, it is likely that a name may be chosen for more than one claim, causing a name conflict. LBRY resolves the conflict by using the LBRY blockchain to allow all LBRY nodes track and stay on the same page regarding all the claims. In particular, through the blockchain, LBRY assigns every claim an ID, knows among all claims for a given name which claim is the "controlling" claim, and of all claims for a given name, LBRY can know their order in terms of their time accepted by the LBRY community or in terms of credits backing them, allowing a user to use all such information to name specific claims. Indeed, LBRY supports several types of names, and we describe what these types are and how name resolution works for each type, as follows.

1) Name without modifier (`lbry://<name>`): This URL only consists of a bare name on its own. If there are more

than one claim with the same name, of all of the claims named <name>, this URL resolves to the controlling claim for the name, which is the claim with the most credits that the publisher of the claim and the entire LBRY community have committed towards it. If it is a stream claim name, it resolves to a controlling stream claim. If it is a channel claim name, the first character of <name> is '@', and it resolves to a controlling channel claim. Such an URL is not permanently owned by a publisher. Instead, it is controlled by the LBRY community itself and may map to a different controlling claim which the community determines at different time. See Section VII for more details about the controlling claim.

2) Name with modifier (lbry://<name><modifier>): The name for a stream claim or a channel claim can be further followed by one of the three different modifiers:

- **Claim ID**: '#' followed by a hexadecimal number (e.g. lbry://name#8 or lbry://name#ab). This URL resolves to a claim for this name with the specific claim ID as defined by the hexadecimal number. The hexadecimal number here can also just be a prefix of the entire claim ID; if more than one claim satisfy the prefix matching, the claim that was created earliest according to the LBRY blockchain will be returned. Such an URL is permanently owned and controlled by the publisher.
- **Sequence**: ':' followed by a positive number n. From the list of all claims for the claim name, this URL will resolve to the nth claim that the LBRY community accepted for this name, according to the LBRY blockchain.
- **Amount Order**: '$' followed by a positive number n. From the list of all claims for the claim name, this URL returns the nth claim for this name ordered by the total amount of credits backing the claims (highest first).

3) Channel claim name and stream claim name (lbry://<@channel_name>/<stream_name>): This URL contains both a channel and a stream claim name, both with or without a modifier. It refers to content published to the name <stream_name> within the channel <@channel_name>. If multiple claims for the same name exist inside the same channel, they are resolved via the same resolution rules as above, but applied entirely within the sub-scope of the channel.

The resolution happens in two steps. First, it resolves the channel to its associated claim: it removes the '/' and <stream_name> from the path, and resolves the URL as if it was lbry://<@channel_name>. Then it resolves the stream claim name to the appropriate claim from among the claims in the channel: it obtains the list of all claims in that channel, and resolves the <stream_name> as if it was its own URL, but instead of considering all claims, it only considers the set of claims in the channel.

VII. LBRY Blockchain

A. Overview

The design of the LBRY blockchain is based on the Bitcoin blockchain, with significant modifications. On one hand, like Bitcoin, the LBRY blockchain is a public, proof-of-work blockchain, and it maintains balances of **LBC**, LBRY's cryptocurrency and unit of credit. On the other hand, different from the Bitcoin blockchain that supports arbitrary transactions, the LBRY blockchain supports a continuously running auction of content names. It supports a specific set of commands that allows anyone to bid (in LBC) to control a LBRY name such that whichever party (or parties) bids the most LBC have the right to use the name for a channel or certain content of theirs and to control the metadata for the name. The LBRY blockchain also uses its consensus mechanism to maintain names and metadata of content and channels available on the LBRY network—i.e., the Claimtrie (Section V-B). As a result, via the LBRY blockchain, whoever controls a name can describe what it contains, what it costs to access, who to pay, and where to find it, and anyone can look up names and their metadata.

The LBRY blockchain further serves as a payment system and record of purchases for priced content. As a LBRY client can easily look up content names and metadata and learn how to access and download content according to the metadata, it can issue payments on the blockchain, or via off-chain settlements, to purchase specific content. The LBRY blockchain supports proof of payment such that a transaction spends a correct amount of credits from a legitimate client to a correct address.

B. LBRY Credits (LBC)

LBRY Credits, or LBC, are the unit of account for LBRY. Eventually 1,000,000,000 LBC will exist, according to a defined schedule over 20 years. The schedule decays exponentially, with around $100,000,000$ in the first year. Additionally, some credits are awarded on a fixed basis.

C. Claim Stake and Support Stake

The LBRY blockchain introduces a new entry type called **stake**. Every stake is to track credits committed toward a stream or a channel. Every stake has an *id* that is a 20-byte hash unique among all stakes and an *amount* that is the number of credits used to back the stake.

There are two ways to compose a stake. One way is to enhance a claim with a stake id and an amount of credits to back the claim, thus making a **claim stake**. (From now on we use the term "claim" and "claim stake" interchangeably.) The id of a claim stake is also called **ClaimID**. The other is to lend certain amount of credits to bolster an existing claim stake that is identified by its ClaimID, which makes a **support stake** with its own id and the support amount. The id of a support stake is also called **SupportID**. Figure 4 is an example stream claim stake. Figure 5 is an example support stake for this claim.

Now that every claim is a claim stake in the LBRY blockchain, a claim not only describes the name and metadata of a particular content or a content publisher identity, but also has credits committed toward it.

Recall every claim has a name and a value, where the value is the metadata of a stream claim or a channel claim. The

```
{
  claimID:    "6e56325c5351ceda2dd0
               795a30e864492910ccbf",
  amount:     100.0,
  name:       "lbry_intro",
  stream: {
    streamHash:  "232068af6d51325c4821ac89
                  7d13d7837265812164021ec8
                  32cb7f18b9caf6c77c23016b
                  31bac9747e7d5d9be7f4b752",
    fee:         ...
    title:       "What is LBRY?",
    description: "What is LBRY? An introduc-
                  tion with Alex Tabarrok",
    language:    "en",
    thumbnail:   "https://s3.amazonaws.com/
                  files.lbry.io/logo.png",
    mediaType:   "video/mp4",
    license:     "Public Domain"
    ...
  },
  in_channel: {
    name:       "@LBRY",
    amount:     70000.0,
    signature: "d0e84c5ceecd2c54e861ebe1
                8e4289f6bb57900d45ac88d6
                 189645d5b9ac87ef44ac967d",
    }
  }
}
```

Fig. 4: **An example stream claim stake.**

```
{
  supportID: fbcc019294468e03a597
             0dd2adec1535c52365e6,
  amount:    500.0,
  claimID:   6e56325c5351ceda2dd0
             795a30e864492910ccbf,
}
```

Fig. 5: **An example support stake.**

blockchain treats every metadata as an opaque series of bytes and stores it in a serialized binary format by encoding it using Protocol Buffers [21]. Such encoding allows for metadata to be extensible (to encompass thousands of fields for dozens of types of content), compact (space-efficient), and interoperable (i.e., usable by many projects written in different languages). The serialized metadata may be cryptographically signed to indicate its ownership. If a metadata belongs to a channel, it can also be signed by the channel's private key. Each client is responsible for correctly encoding and decoding the metadata, and for validating its structure and signatures.

D. LBRY Transactions

Like the Bitcoin blockchain, the core operation of the LBRY blockchain is to create, propagate, validate, and finally add transactions to the blockchain which is a global ledger of transactions. Every transaction includes inputs that specify which LBC credits to consume and outputs that are chunks of LBC credits to be validated and recorded. Same as Bitcoin, LBRY also uses the stack-based transaction script language, *Script*, to embed instructions in a transaction to describe how to handle the transaction, including allowing a sender to specify conditions that a recipient has to meet in order to spend the output of the transaction. Besides the Script words, also known as **opcodes** or commands, that Bitcoin supports[1], LBRY introduces three new opcodes to enable transactions that interact with and change the state of the Claimtrie:

1) **OP_CLAIM_NAME**: Creates a new claim on the Claimtrie.
2) **OP_SUPPORT_CLAIM**: Supports a claim by adding more credits toward it.
3) **OP_UPDATE_CLAIM**: Updates the metadata of a claim.

More specifically, every new opcode can be used in the outputs of a transaction, with the opcode followed by one or more parameters, as follows (we skip the rest of the script related to specifying the spending condition of the transaction):

1) `OP_CLAIM_NAME <name> <value> ...`: This transaction creates a new claim stake on the Claimtrie. Its name is `<name>` and its metadata is decoded from `<value>`. The outputs of this transaction also defines the credits committed to the claim stake, i.e., its amount.
2) `OP_SUPPORT_CLAIM <name> <claimID> ...`: This transaction creates a new support stake that supports the claim on the Claimtrie whose name is `<name>` and whose ClaimID is `<claimID>`. The outputs of this transaction also defines the credits to bolster the claim.
3) `OP_UPDATE_CLAIM <name> <claimID> <value> ...`: This transaction updates the claim on the Claimtrie whose name is `<name>` and whose ClaimID is `<claimID>` with new metadata decoded from `<value>`. Note this transaction is required to spend the early transaction output for the claim; otherwise it is considered invalid.

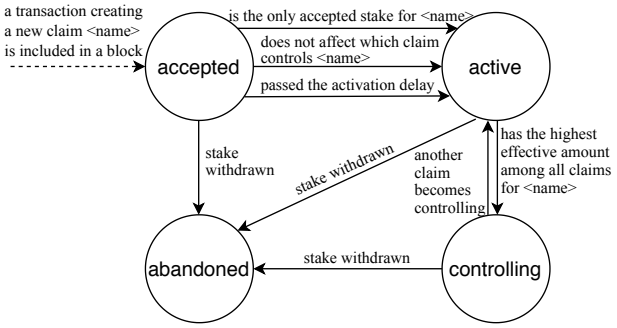
The `<value>` above is the encoded byte stream of claim metadata and optional channel signature (Section VII-C)).

E. Stake Status

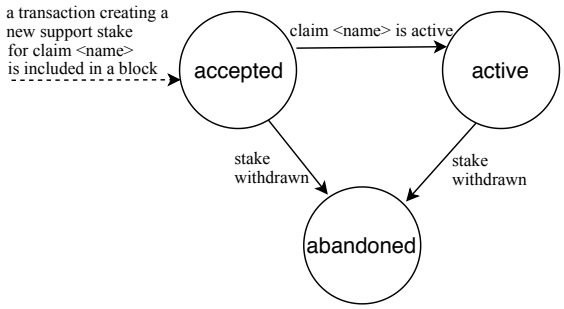
Driven by these operations, a stake can have one of four statuses in a given block: **accepted**, **active**, **controlling (claims only)**, and **abandoned**.

Figure 6 is a finite-state machine that describes how the status of a stake changes. When a transaction that creates a new claim or support stake is included in a block, thus the stake is entered into the LBRY blockchain, the stake is then **accepted**. Here, the sum of the amount of a claim stake and all of its accepted supports is called its **total amount**. After a stake is accepted, if the stake is the only accepted claim for a name, or it does not affect which claim controls the name (see below), the stake becomes **active** immediately. Also, if the stake is a support to an *active* claim, it also becomes active immediately. Otherwise, the stake needs to be in the blockchain for an algorithmically determined number of blocks—which is called

[1]see https://en.bitcoin.it/wiki/Script



(a) Claim stake <name>



(b) Support stake for claim <name>

Fig. 6: **Finite-state machine of stake.**

an **activation delay**—before its status becomes *active*. For an active claim, the sum of the amount of the claim itself and all of its active supports is called its **effective amount**. (Claims that are not active have an effective amount of 0.)

As multiple claims can compete for the same name and thus stored in the same leaf node for that name, they are sorted and the active claim that is first in the sort order *controls* the name; this claim is also called a **controlling** claim. For a given name at a given block, only one claim can be controlling, and the sorting is primarily based on the effective amount of all these claims, as we designed in the following algorithm:

1) Calculate the effective amount of every *currently* active claim.
2) Sort the claims by effective amount in descending order. If claims are tied for the same amount, order the claims by the height of the blocks in which the claims are created, with the one at the lowest height first; if they are created in the same block, order them by their transaction order within the block.
3) If the controlling claim from the previous block is no longer the first in the order, such as after a support to an active claim that increases the effective amount of the claim, invoke a **takeover** process. The takeover process sets the takeover height for this name to the current block height, which may change the activation delay of certain accepted claims (see below) and make them active, thus adding more active claims to compete for the control of the name. Recalculate which stakes are now active, and go back to Steps 1 and 2.
4) The ordering is finished; the claim with the greatest effective amount is the controlling claim at this block.

Block 13: Claim A for 10LBC is accepted. It is the first claim, so it immediately becomes active and controlling.
State: A(10) is controlling

Block 1001: Claim B for 20LBC is accepted. Its activation height is `1001 + min(4032, floor((1001-13) / 32)) = 1001 + 30 = 1031`.
State: A(10) is controlling, B(20) is accepted.

Block 1010: Support X for 14LBC for claim A is accepted. Since it is a support for the controlling claim, it activates immediately.
State: A(10+14) is controlling, B(20) is accepted.

Block 1020: Claim C for 50LBC is accepted. The activation height is `1020 + min(4032, floor((1020-13) / 32)) = 1020 + 31 = 1051`.
State: A(10+14) is controlling, B(20) is accepted, C(50) is accepted.

Block 1031: Claim B activates. It has 20LBC, while claim A has 24LBC (10 original + 14 from support X). There is no takeover, and claim A remains controlling.
State: A(10+14) is controlling, B(20) is active, C(50) is accepted.

Block 1040: Claim D for 300LBC is accepted. The activation height is `1040 + min(4032, floor((1040-13) / 32)) = 1040 + 32 = 1072`.
State: A(10+14) is controlling, B(20) is active, C(50) is accepted, D(300) is accepted.

Block 1051: Claim C activates. It has 50LBC, while claim A has 24LBC, so a takeover is initiated. The takeover height for this name is set to 1051, and therefore the activation delay for all the claims becomes `min(4032, floor((1051-1051) / 32)) = 0`. All the claims become active. The totals for each claim are recalculated, and claim D becomes controlling because it has the highest total.
State: A(10+14) is active, B(20) is active, C(50) is active, D(300) is controlling.

Fig. 7: **A step-by-step example of how competing claims activate and are ordered.** All stakes are for the same name.

Imposing an *activation delay* on an accepted stake gives long-standing claimants time to respond to changes, while still keeping takeover times reasonable and allowing recent or contentious claims to change state quickly. The delay is determined by a formula:

$$min(4032, \lfloor (H_{Accept} - H_{Takeover})/32 \rfloor)$$

where H_{Accept} is the height of the block in which the stake was accepted and $H_{Takeover}$ is the height of the most recent block at which the controlling claim for the name is defined or changed. With 2.5 minutes per block (the target block time), it takes 224 days without a takeover to reach the max delay.

Figure 7 shows a step-by-step example of how competing claims activate and are ordered.

Finally, a stake can also be **abandoned** and removed from the Claimtrie. The owner of a stake can withdraw the stake by spending a transaction that contains the stake, thus causing that stake to become abandoned.

F. Claimtrie Consensus

As claims are stored at the leaf nodes of the Claimtrie, LBRY further derive from these leaf nodes a Merkle hash tree, including obtaining the hash of the root node of the Merkle hash tree—which is also called the **root hash**. For each block in the LBRY blockchain, its block header stores the root hash of the Claimtrie at that block height, enabling any LBRY node to use the root hash to efficiently and securely validate the state

of the Claimtrie. As the root hash is derived from all the leaf nodes using the Merkle hash tree algorithm, if any LBRY blockchain network node disagrees that a claim name was accepted or who is the controlling claim of a name etc., its root cache will differ and the block will not form the same chain as the ones that do agree. This is the same for the traditional Merkle hash tree root formed by transactions in a block.

G. Purchasing Content

Once a user decides to access and purchase a specific content (such as in Step 5 of the sample use in Section III), the LBRY client of the user will look up for the name associated with the content in order to acquire the metadata associated with the name. If the client has a local copy of the Claimtrie, it will look up for the name in the Claimtrie; otherwise, it broadcast this lookup to miners or a service provider which has a local copy of the Claimtrie. Once the lookup is successful, the client then checks any required payments and acts accordingly, as instructed by metadata entries:

1) If the content is set to free, the client uses the metadata to start downloading the content.
2) If the content is set to have a price in LBC, the client must issue a payment in LBC to the specified address before downloading the content.
3) If the content is published encrypted, the client can uses the metadata to start downloading the content, but meanwhile it must issue payments to receive the key in order to decrypt the downloaded content.
4) If the content is set to have another payment method, the seller must run or use a service that provides a private server enforcing payment and the provisioning of accessing keys.

While payments can be issued directly on the LBRY blockchain, the LBRY protocol encourages a volume of transactions that will not scale without usage of off-chain settlement. Essentially, rather than issue a transaction to the core blockchain, transactions are issued to a 3rd-party provider. These providers have a substantial number of coins which are used to maintain balances internally and settle a smaller number of transactions to the core chain. In exchange, these providers earn a small fee, less than the fee required to issue the transaction directly to the blockchain.

Another addition of the LBRY blockchain on top of the Bitcoin is **proof of payment**. A proof of payment has two components:

- A transaction on the blockchain that spends credits to the fee address for a claim (the transaction must send a number of credits equal to or greater than the fee amount for the claim).
- Proof that a client knows the private key of the address that the transaction spends from.

To prove 1, it is sufficient to provide the transaction ID and input index of the spend. Proving 2 requires signing a nonce using the associated private key.

Verifying a proof of payment is done as follows:

- Look up the fee amount and fee address of the claim that the proof is for.
- Use the transaction ID from the proof to find the transaction. Verify that it spends the correct amount to the correct address.
- Use the public key from the transaction output to verify the signed nonce.

The current version of the protocol does not support sophisticated price negotiation between clients and hosts. The host simply chooses the price it wants to charge. Clients check this price before downloading and pay the price after the download is complete. Future protocol versions will include more options for price negotiation, as well as stronger proofs of payment.

VIII. Running LBRY

We have been running LBRY since June 2016. Figure 8 shows the number of LBRY users who perform at least one metadata resolution per month. Figure 9 shows the data throughput per day. As the Claimtrie is the core data structure of LBRY, Figure 10 shows the size of the Claimtrie as the blockchain grows. Below we discuss several important issues in running LBRY.

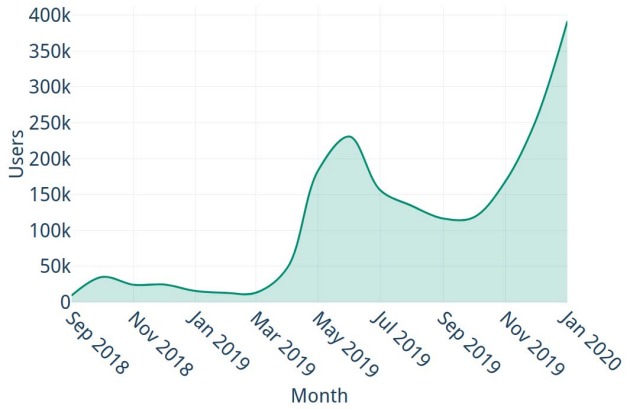


Fig. 8: **# of users who perform at least one metadata resolution per month.**

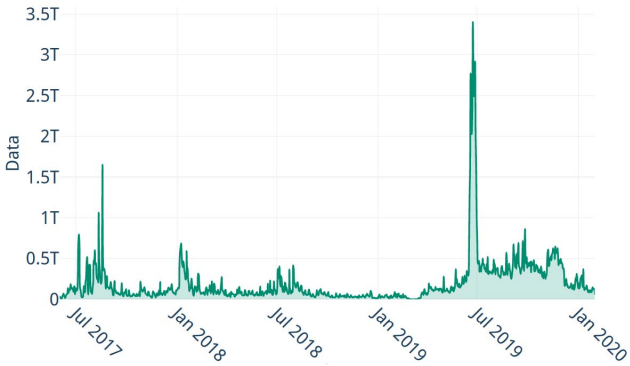


Fig. 9: **Estimated throughput per day (MB).**

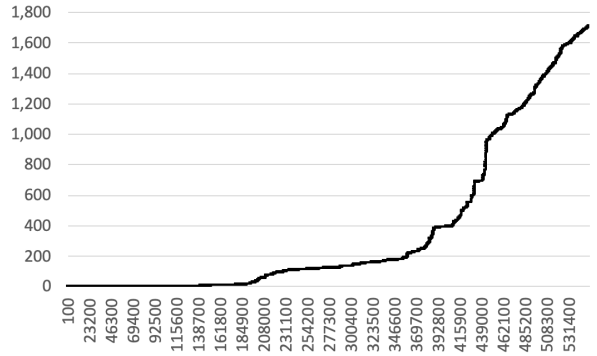


Fig. 10: **Claimtrie memory size (MB) at different block height.**

A. Content Discovery

Although the name space provided by LBRY is helpful towards discovery, much as the web would be much less useful without search engines or aggregators, LBRY needs its own discovery mechanisms.

Search features can be constructed from the catalogue of metadata provided in the blockchain as well as the content transaction history available in the blockchain or observed on the network. All of this data, along with user history, allows for the creation of content recommendation engines and advanced search features.

Discovery on LBRY can also take the form of featured content. Clients can utilize featured content to provide additional visibility for new content that consumers might not otherwise be looking for.

B. Content Distribution

Digital content distributors with server-client models are subject to the whims of Internet service providers and hostile foreign governments. Traffic from the host servers can be throttled or halted altogether if the owners of cables and routers so choose. However, in the case of LBRY, content comes from anywhere and everywhere and is therefore not so easily stifled.

Additionally, the market mechanisms of LBRY create a strong incentive for efficient distribution, which will save the costs of producers and ISPs alike. These properties, along with LBRY's infringement disincentivizing properties, make LBRY an appealing technology for large existing data or content distributors.

IX. Conclusions

Despite that the need for users to publish, host, find, download, and pay for content online is only growing, the common solutions users have are not satisfactory. They either have to use a centralized platform, thus subject to their terms and practice, or use a peer-to-peer solution such as BitTorrent where one has to face a daunting task of discovering the hash of the content they seek. This paper offers a new solution, LBRY, that allows users to create, name, publish, search, access, download, and purchase their content through a decentralized content platform, all with ease. LBRY introduces a new naming scheme that gives users the full control of the names of their content, and uses a blockchain to not only support a digital currency (LBC) and transparent decentralized ledger, but also allow every user to access a synchronized name space and a global index of content metadata, thus supporting a new paradigm of digital content distribution and marketplace.

References

[1] B. Cohen, "The bittorrent protocol specification," 2008.
[2] LBRY Inc. (2019) LBRY app download. https://lbry.io/get.
[3] S. Saroiu, K. P. Gummadi, R. J. Dunn, S. D. Gribble, and H. M. Levy, "An analysis of internet content delivery systems," *SIGOPS Oper. Syst. Rev.*, vol. 36, no. SI, pp. 315–327, Dec. 2002. [Online]. Available: http://doi.acm.org/10.1145/844128.844158
[4] A.-M. K. Pathan and R. Buyya, "A taxonomy and survey of content delivery networks," *Grid Computing and Distributed Systems Laboratory, University of Melbourne, Technical Report*, vol. 4, no. 0, p. 1, 2007.
[5] N. Anjum, D. Karamshuk, M. Shikh-Bahaei, and N. Sastry, "Survey on peer-assisted content delivery networks," *Computer Networks*, vol. 116, pp. 79–95, 2017.
[6] R. Rodrigues and P. Druschel, "Peer-to-peer systems," *Commun. ACM*, vol. 53, no. 10, pp. 72–82, Oct. 2010. [Online]. Available: http://doi.acm.org/10.1145/1831407.1831427
[7] S. Nakamoto *et al.*, "Bitcoin: A peer-to-peer electronic cash system," 2008.
[8] S. Fujimura, H. Watanabe, A. Nakadaira, T. Yamada, A. Akutsu, and J. J. Kishigami, "Bright: A concept for a decentralized rights management system based on blockchain," in *2015 IEEE 5th International Conference on Consumer Electronics-Berlin (ICCE-Berlin)*. IEEE, 2015, pp. 345–346.
[9] D. Bhowmik and T. Feng, "The multimedia blockchain: A distributed and tamper-proof media transaction framework," in *2017 22nd International Conference on Digital Signal Processing (DSP)*. IEEE, 2017, pp. 1–5.
[10] M. Fukumitsu, S. Hasegawa, J. Iwazaki, M. Sakai, and D. Takahashi, "A proposal of a secure p2p-type storage scheme by using the secret sharing and the blockchain," in *2017 IEEE 31st International Conference on Advanced Information Networking and Applications (AINA)*. IEEE, 2017, pp. 803–810.
[11] J. Constine, "Monegraph uses bitcoin tech so internet artists can establish 'original'copies of their work," 2016.
[12] S. Wilkinson, T. Boshevski, J. Brandoff, and V. Buterin, "Storj a peer-to-peer cloud storage network," 2014.
[13] S. Wilkinson, J. Lowry, and T. Boshevski, "Metadisk a blockchain-based decentralized file storage application," *Storj Labs Inc., Technical Report, hal*, pp. 1–11, 2014.
[14] T. McConaghy, R. Marques, A. Müller, D. De Jonghe, T. McConaghy, G. McMullen, R. Henderson, S. Bellemare, and A. Granzotto, "Bigchaindb: a scalable blockchain database," *white paper, BigChainDB*, 2016.
[15] author. (2015) Canonical JSON. http://wiki.laptop.org/go/Canonical_JSON.
[16] P. Maymounkov and D. Mazieres, "Kademlia: A peer-to-peer information system based on the xor metric," in *International Workshop on Peer-to-Peer Systems*. Springer, 2002, pp. 53–65.
[17] Internet Assigned Numbers Authority (IANA). (2019) Media types. https://www.iso.org/iso-639-language-codes.html.
[18] ——. (2019) Media types. https://www.iana.org/assignments/media-types/media-types.xhtml.
[19] P. E. Black. (2019) Trie. https://xlinux.nist.gov/dads/HTML/trie.html.
[20] M. Davis and K. Whistler. (2018) Unicode normalization forms. https://www.unicode.org/reports/tr15/tr15-47.html.
[21] Protocol Buffers. https://developers.google.com/protocol-buffers.