A Robot Economy for Music Without Any Intermediaries

Tim Wissel

A Robot Economy for Music Without Any Intermediaries

by

Tim Wissel

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on XXX at XXX.

Student number:	4460251	
Project duration:	March 6, 2020 – XXX	
Thesis committee:	Dr. ir. J. Pouwelse,	TU Delft, supervisor

Preface

For my master thesis, I was interested in exploring an area with present-day social issues and in building real applications. I am a big fan of music, and especially of artists in some more underground/unknown genres. Ever since I started paying for a streaming service subscription, I wondered what would happen with this money. I also wanted to support my favorite artists directly but found that there was no easy way to do this, and that some party would always get an unfair share of my money. This sparked my interest for a topic that J. Pouwelse discussed with me, that came to be my thesis topic.

I want to thank my supervisor J. Pouwelse for giving many inspiring ideas, and for giving me the freedom to explore and expand my own visions. I want to thank all PhD and Master students of the distributed systems group for enthusiastic explanations of their techologies. I want to thank in particular Matt Skala, for being very helpful to me when I started building the foundations of my application.

Tim Wissel
Delft, October 2020

Contents

1 Introduction

Music streaming services have an immense amount of power in the music industry. The top streaming services dictate the rules which artists have to play by. The top 5 streaming services form an oligarchy, and the same goes for the top 3 labels. This leads to a centralization of power, and problems related to IT gatekeeping and platformization.

1.1. Centralization of power

Firstly, corporations with power squeeze the music production side by taking large cuts of revenue from the user subscription money. As a result, the artists receive a low compensation. Especially independent artists have a hard time making a living. The distributors Spotify, iTunes and Google Play take on average a 25% cut for signed records and 40% cut for unsigned records.

Secondly, Big Tech has curatorial power to decide what is shown in the catalog of their application. The music catalog may seem endless, but in reality it is controlled by the Big Tech corporation and dictated by the interests of major labels. The inner workings of recommendation algorithms and playlists are in the hands of a few labels and streaming services.

Finally, the streaming oligarchy can sensor tracks. The freedom of artist expression is then decided by undemocratic judgments. Big Tech has the power to decide the future of an artist.

1.2. Proposed solution: MusicDAO

This thesis proposes an alternative technology from Big Tech streaming services. We design and implement a decentralized system which attempts to replace the full value chain in music streaming industry, from the subscription money to the artist, by removing all intermediaries and giving power back to the artists and listeners. Listeners can stream music without being dependent on a single provider and can give money directly to artists. Artists receive 100% of this donation and subscription money.

In essence, the solution is a decentralized autonomous organization (DAO) which is formed by listeners and artists. In this paper we present the design and implementation of our mobile android app MusicDAO. Users of this app form a phone-to-phone serverless network they form a network over which they publish music, download music and transfer money. Peer-to-peer communication and peer-to-peer payments are essential technologies for this. No external servers, third parties or intermediaries are necessary to keep this flow of music and money going. Any person can join the network, publish music and get paid for it.

MusicDAO supports the following functionalities:

- Defining and publishing music content with metadata;
- Streaming music over BitTorrent;
- Caching and streaming optimization algorithms;
- Browsing playlists;
- Remote keyword search;
- Peer-to-peer donations to artists using Bitcoin.

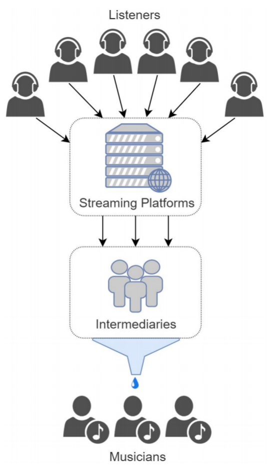


Figure 1.1: Artist compensation inconsistency

1.2.1. Experimentation and evaluation

In a real-world experiment with Android phones, we tested the feasibility of such a phone-to-phone serverless system. We ran MusicDAO on at least X android devices, and registered the latency of retrieving music metadata, and transaction speeds of transferring money and audio files.

2
Problem description

Music artists have a hard time making a living. The oligarchical power of music streaming services and labels squeeze the production size of the music industry. The most widely used music streaming services, with the largest music catalogs, run centralized, proprietary and closed-source software. The companies owning these services have a large amount of power in the music streaming industry because of their scale. The top 5 streaming services have a combined market share of 81%, so this can be regarded as an oligarchy. Because of their power, they can ask high commission fees or lock artists to one platform. As a result, artists receive low compensation. Furthermore, the recommendation and playlist generation algorithms are also a black box for the user. This gives streaming companies curatorial power. At the time of writing, there exists no alternative decentralized and transparent music streaming system with peer-to-peer payments directly to artists. A visualization of the economic muscle of both the label oligarchy and streaming platform oligarchy is shown in 2.1. *How can we design and implement a music streaming service that distributes the power from one authority to its users?*

2.1. Platformization and gatekeeping

The infrastructure of current internet applications are increasingly moving towards 'platformization'. In essence, platforms are taking control of "the surface on which the market exchange take place" (Andersson Schwarz, 2016) with digital distribution and network effects enabling an increasing centralization of power. This phenomenon is related to IT gatekeeping: tying access of content to a specific internet service. An example of this is the release of the album *The Life of Pablo* in 2016, which was contracted to only be played on one platform, Tidal. In addition, the big music platforms are able to create a digital enclosure of content. This facilitates rent-based monetization of user data, described by Harvey (2002) as 'monopoly rent'. Namely, advertisers can lease a virtual space, to match their advertisements to user profiles. The latest movement in platform accumulation is the monopolization of data. Large scale of data about user interactions with the platform forms a 'monopoly of knowledge' (Innis, 2007). The power of platform companies are raising because platforms, in general, tend towards monopoly (Srnicek, 2017).

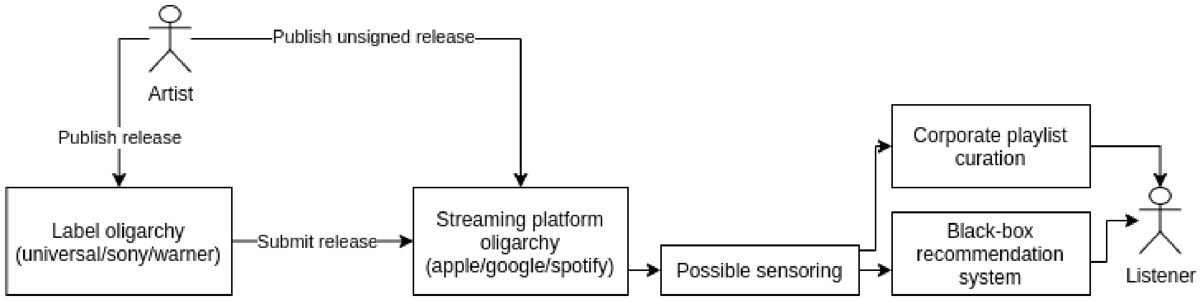

Artist
Publish unsigned release
Publish release
Label oligarchy (universal/sony/warner)
Submit release
Streaming platform oligarchy (apple/google/spotify)
Possible sensoring
Corporate playlist curation
Black-box recommendation system
Listener

Figure 2.1: The current flow from publishing music to receiving it as a listener. The figure shows that the framework on which music is published and found is dominated by label and streaming oligarchs and by corporate decisions.

Platform	**Music release type**	**Label cut**	**Platform cut**	**Artist/band cut**	**Streams per month to earn min. wage (solo musician)**
Spotify	Unsigned	0%	40%	60%	180,000
	Signed	55%	25%	20%	1,117,02
Apple Music	Signed	55%	25%	20%	971,323
Google Play	Unsigned	0%	40%	60%	70,391
	Signed	55%	25%	20%	172,206

Table 2.1: Estimated revenue cuts on streaming platforms, with a calculation on the streams/plays per month that an artist should have in order to make a minimum wage (SNEP, 2015)

In relation to gatekeeping, platforms are now given the task to perform moral judgements on content, for example whether to censor a certain artist. This is controversial as these judgements are no longer in the hands of users but rather in the hands of companies. Recent examples exist such as the disappearance of Li Zhi[1], who published songs about democracy and social issues in China. All of China's main streaming sites removed his songs. In 2019, Apple Music also removed content from their platform by singer Jacky Cheung, who referenced the tragedies of Tiananmen Square in his songs[2].

2.2. Intermediaries take a large share

Artists publishing their content on a music streaming service such as Google Music, Spotify and Apple Music receive low compensation, because the intermediaries take a large cut in revenue, typically between 25 and 40 percent (see 2.1). According to Midia Research, the top 5 streaming services control 81% market share (Mulligan, 2020), which can be regarded as an oligarchy. Together these streaming services have the power to ask high subscription fees. After multiple consecutive years of growth, 2019 became the first year in which digital streaming is the single biggest source of revenue for the music industry globally (IFPI, 2020). The global music market also gained a 33.5% growth in paid streaming subscribers during this year (IFPI, 2020). This trend is shifting power from decentralized music stores to centralized streaming services.

At the same time, streaming services take a large cut in revenue, and artists are having a harder time making money from music. According to investigations by aCooke (2018) and ReCode (2015), the revenue cut of Apple Music and Spotify is between 25% and 30%. An additional problem is opacity: streaming deals on these platforms remain behind closed doors. For these reasons, massive audiences are needed to generate sustaining profits. An investigation by Bloomberg[3] shows that a whopping 152,094 Spotify subscriber streams generate $100 on average for artists. Consequently, only 0.733% of all acts generate enough revenue for an artist to make a living (Ingham, 2018). The International Federation of the Phonographic Industry states that as of 2018 there exists a "value gap" in digital music streaming, meaning a "mismatch between the value that some digital platforms [...] extract from music and the revenue returned to the music community–those who are creating and investing in music" (IFPI, 2018).

2.3. Monopsony power

Monopsony power means that a dominant buyer has the power to push prices down with suppliers. In the context of music, this means that artists have little choice over which platform to publish their music on, because of the dominance of one platform. A few major players in the music industry together form an oligopolistic market. Monopsony power in this area can lead to squeezing the producer side. An example of monopsony power is an event that happened in 2014, between Amazon and Hachette. Amazon, having a large market share on e-books, used its commercial muscle to demand a larger cut of the price of Hachette books it sells. This included for all Hachette books "preventing customers from being able to pre-order titles, reducing the discounts it offered on books and delaying shipment" (Ellis-Petersen, 2014).

Along the same lines, the music streaming oligarchs can use their commercial muscle to demand low pays to artists. Spotify founder and CEO Daniel Ek declared to its investors that the increase in interactions with

[1]https://www.independent.co.uk/news/world/asia/tiananmen-square-china-li-zhi-singer-disappears-anniversary-protests-a8940641.html

[2]https://hongkongfp.com/2019/04/09/apple-music-china-removes-jacky-cheung-song-reference-tiananmen-massacre/

[3]https://www.bloomberg.com/opinion/articles/2017-09-25/the-music-business-is-more-unfair-than-ever

its in-house curated playlists “puts Spotify in control of the demand curve” [4].

2.4. Recommendations and curatorial power

The Big Tech music companies recommend content that best fits their business model, which may be contrary to what is most useful to their customer. The companies can promote or dis-promote content by their choosing. This shows “curatorial power”: the ability to advance own interests by organizing and prioritizing content (Prey, 2020).

Musicians and record labels are increasingly more dependent on landing on Spotify-curated playlists. For example, a study done by the European Commission shows that, for a track to land on the Spotify-curated playlist “Today’s Top Hits”, it will see an increase of $163,000 in revenue (Aguiar & Waldfogel, 2018). 99 of the top 100 playlists are curated by the streaming company. So its recommendation algorithms and playlist curation systems are highly influential.

2.4.1. Black box

As the top music streaming services run closed-source software, the inner workings of the recommendation algorithms are opaque to the user. These algorithms, fed by user interaction data, are in some extend also a black box to the company, as they are built using machine learning technologies. However, as noted by (Gillespie, 2014), the impression that algorithms are objective is a “carefully crafted fiction”. Namely, they are altered based on company strategies. Companies are not obliged to explain their algorithms. In the context of recommendations, this leads to a “threat of invisibility": the problem of content regularly disappearing (Bucher, 2018), a phenomenon which is out of the hands of the artist, because of an asymmetry in knowledge over the algorithm workings. Frustrating for artists and labels is also the opaqueness of getting playlisted: it is unclear why “[...] a particular track was placed, or replaced, on a playlist” (Prey, 2020).

2.4.2. Single point of responsibility

On the other hand, if the ‘frontpage’ playlists, such as “Today’s Top Hits”, are manually created by a person or company, we run into another issue. This is explained in a recent book by Heuvelings (2020). The author had the job of maintaining several high-demand playlists on Spotify, with millions of monthly listeners. This gave her substantial power but also huge pressure from artists and labels, demanding their work to be visible on her playlists. She was threatened by some of these parties as well, after which she decided to leave Spotify.

This autobiography shows the immense pressure on playlist curation, when this is done by one company or person. It can make or break an artist, so there will be large (financial) pressure from labels or artists towards playlist maintainers, making the music industry unfair for smaller artists with less power.

[4]`https://investors.spotify.com/financials/press-release-details/2019/Spotify-Technology-SA-Announces-Financial-Results-for-Fourth-Quarter-2018/default.aspx`

3 State of the Art

In this chapter we describe existing algorithms, protocols and applications in computer science, which try to solve, or give an alternative for, the centralized power in digital audio streaming. We inspect full solutions in the form of distributed applications, and techniques which solve subproblems, such as decentralized file sharing protocols.

3.1. Decentralized music distribution technologies

Multiple decentralized audio distribution and streaming applications exist. Examples are Audius (Rumburg, Sethi, & Nagaraj, 2018), Musicoin [1] and Opus Audio (Jia et al., 2016).

3.1.1. Audius

Audius (Rumburg et al., 2018) presents a decentralized protocol for audio content, which aims to improve payouts to artists and its transparency. It contains a token economy with a transparent payout system for the artists, and a user-operated, distributed network for metadata and content. In addition, it has a governance system like a DAO, in which users can decide on changes to the protocol by democratic voting. Its protocol is established around the ideology of disintermediation: "Intermediaries should be removed when possible; when necessary, they should be algorithmic, transparent, and verifiably accurate" (Rumburg et al., 2018). It uses IPFS (Benet, 2014) for storage of audio content, meaning it relies on voluntarily-hosted high-performant servers.

3.1.2. Opus Audio

Opus Audio (Jia et al., 2016) is a decentralized music-sharing platform and proposes a solution for music ownership registration on a blockchain structure. It has a running distributed audio file sharing system using the Inter-Planetary FileSystem designed by Benet (2014) (see 3.5). It contains a decentralized and fully automated system for purchasing access to music, which works as follows. Opus stores encrypted audio files on a swarm of connected nodes. The decryption keys and files hashes are stored in a smart contract (see 3.2). Using cryptocurrency, users spend their funds on these smart contracts, to unlock access to audio files.

3.1.3. Musicoin

Musicoin is a blockchain platform without intermediaries that focuses on income for independent artists. It uses smart contracts and cryptocurrency to show transparency in payments (see 3.2). This payment structure ensures that each contributor in the network is rewarded, and that artists receive a stable income based on the Universal Basic Income ideology. Aside from their blockchain architecture, they introduce the $MUSIC cryptocurrency, which they invision as the currency for all economic activity in the music industry. Not all of their architecture is decentralized. They use centralized registration system for artists and listeners. They pay artists using their $MUSIC currency, of which the value may be highly unstable.s

None of the state-of-the-art decentralized audio streaming technologies show a running, fully decentralized infrastructure with stable income for artists. All of these systems have in common that they save

[1] `http://musicoin.org/`

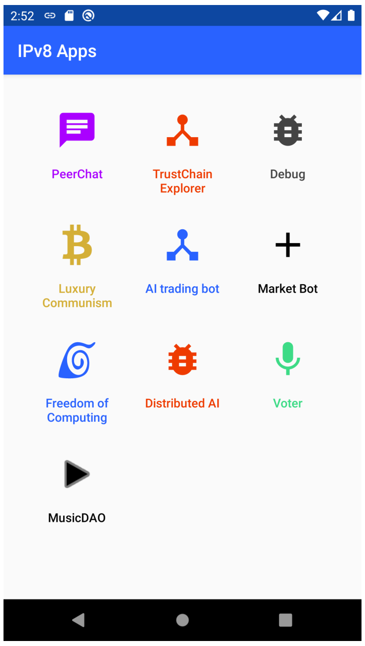
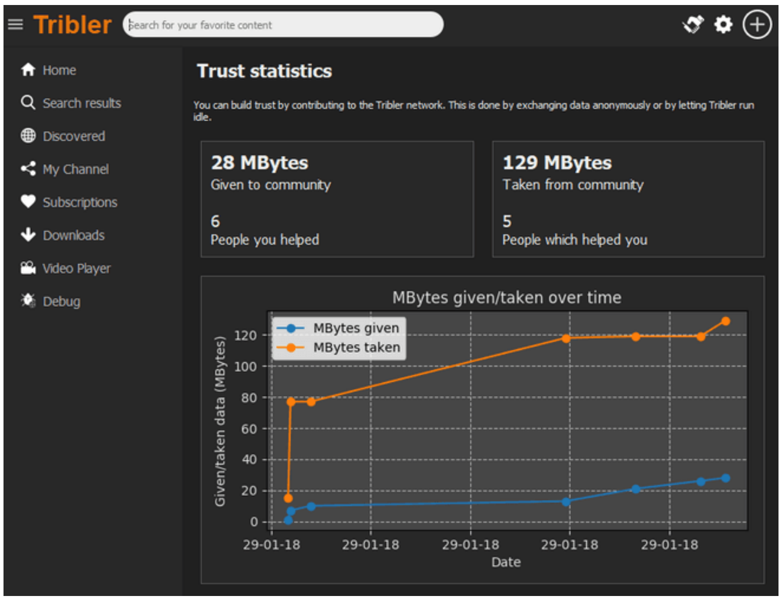


Figure 3.1: Trustchain-Superapp overview



Trust statistics

You can build trust by contributing to the Tribler network. This is done by exchanging data anonymously or by letting Tribler run idle.

28 MBytes
Given to community

6
People you helped

129 MBytes
Taken from community

5
People which helped you

Figure 3.2: Tribler desktop interface, showing the bandwidth incentive system overview

metadata and identifiers of audio files on a blockchain, and save the audio files in an off-chain database using IPFS (Benet, 2014). This makes these solutions reliant on people voluntarily running IPFS content nodes (servers hosting the audio files). In a fully decentralized network, every participant should have the same role, meaning that every node both uploads and downloads content, and it should not be reliant on external servers. Most decentralized systems use their homebrew cryptocurrency to pay artists, instead of a well-established currency or stablecoin. Opus Audio has a development & marketing fund which take a cut of the music revenue, so not all of the revenue goes directly to artists.

3.2. Transparent, automated royalty distribution

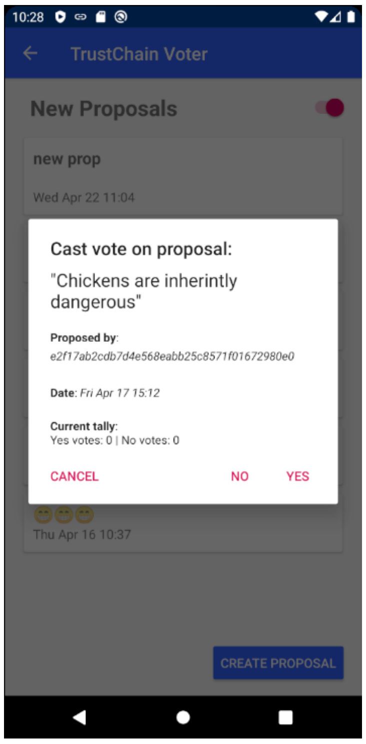
The payment to royalties to artists can be described in a transparent and immutable record on a blockchain. In addition, smart contracts can be used to automate payments (Buterin et al., 2014). Smart contracts are self-executing (no ambiguity) and self-verifying (guaranteeing its statements). In the music industry context, a smart contract can be used for transparent, immutable and automatic payment distribution of royalties. This technique was shown in practice in 2017, when Imogen Heap released the song ‘Tiny Human’ (Heap, 2017). Its distribution of payments to the makers and recorders was written in a smart contract, in the form of a record on the Ethereum blockchain. When a user downloads the corresponding track and makes the payment using cryptocurrency, the forwarding details of the payment are located within the blockchain, and executed as declared on the smart contract.

3.3. Decentralized application frameworks

The TrustChain superapp (Skala, 2020) is a framework for implementing mobile Android decentralized applications. It allows for storing append-only immutable data on TrustChain (Otte, de Vos, & Pouwelse, 2017) and spreading this data in a phone-to-phone serverless network. It follows the concept of super apps (Huang, 2019), meaning that it contains many mini-apps which use the same networking interface. The Superapp is an Android app as seen in fig. 3.1. Its mini-apps implement decentralized democratic voting and has bitcoin payment integration, among other features.

3.4. DAO theory and technologies

Important groundwork around the theory and implementation of a DAO has been done by Jentzsch (2016). He notes that corporations originally work through people only, and this has two flaws: “People do not always


10:28

TrustChain Voter

New Proposals

new prop

Wed Apr 22 11:04

Cast vote on proposal:

"Chickens are inherintly dangerous"

Proposed by: *e2f17ab2cdb7d4e568eabb25c8571f01672980e0*

Date: *Fri Apr 17 15:12*

Current tally:
Yes votes: 0 | No votes: 0

Figure 3.3: Distributed democratic voting mini-app, showing proposals

follow the rules, and they do not always agree what the rules require". His paper illustrates a method that allows creating and maintaining organizations in which "(1) participants maintain direct real-time control of contributed funds and (2) governance rules are formalized, automated and enforced using software".

For an application to evolve without a company or a responsible group of people, decisions need to be made on protocol or feature additions and changes. We inspect the state-of-the-art of peer-to-peer technologies which enable autonomous communities to solve these problems by organizing themselves. In computer science, one approach to this is democratic voting based on voluntary proposals to change and improve the decentralized system.

Jentzsch (2016) proposes a system based on the Ethereum blockchain, using smart contracts (Buterin et al., 2014). In essence, any person that would like to join a DAO must be invited or pay a fee. All participants use tokens, to create proposals and to vote on proposals, so it uses a proof-of-stake mechanism. The paper also contains a solution for the majority owns minority attack: (an attacker with more than half of all tokens can make and accept a proposal to send all its tokens to itself) by allowing participants to split a DAO.

One mobile implementation of this is a mini-app of the aforementioned TrustChain-Superapp (see fig. 3.3). In this voting app, any participant of the organization can create a proposal for the community to vote on. Once a preset voting threshold is reached, the proposal is automatically accepted or denied. Bookkeeping of these proposals is done using the TrustChain distributed ledger technology (Otte et al., 2017). This voting app is an important basis for democratic decision making, but does not include code changes directly after proposals are accepted, so lacks app evolution. This protocol is based on proof-of-identity instead of proof-of-stake, as no tokens are involved.

3.5. Decentralized content delivery networks

A fully decentralized audio streaming service requires sharing and streaming audio files over a network of nodes in which any participant can start and run a node. Well-established examples of such networks are BitTorrent and IPFS.

3.5.1. BitTorrent

BitTorrent (Cohen, 2008) is an open peer-to-peer file sharing protocol. Any person can join the network and start sharing files, by making a torrent which contains the metadata of the files. Content is specified using

SHA1 hashes of the files. Checksums are used to make sure no changes to the original torrent have been made upon receiving files. This way all files are immutably shared. BitTorrent has implementations for various systems, including Android. This means mobile devices can participate in uploading and downloading content, and can even build an autonomous phone-only file sharing system.

Torrent files contain a list of chunks (torrent pieces), which represent the different parts of the related file. Flawless streaming of media files over BitTorrent requires a smart algorithm to predict what file is requested next, and what torrent pieces should be loaded. BitTorrent originally relied on trackers to perform peer discovery, and trackers can become a central point of failure. However, since the introduction of the DHT protocol (Loewenstern & Norberg, 2008), finding peers in the network can be done by querying any known peer, which makes the network more decentralized.

There are no differences between hosters and downloaders. Meaing, all participants have the same capabilities, so every user of the network can both download and upload content. A challenge of the BitTorrent protocol without extensions is incentivizing participants to host files. There are multiple solutions to this problem, discussed in 3.6.

3.5.2. IPFS

The Inter-Planetary FileSystem, introduced by Benet (2014), is a distributed peer-to-peer file sharing system in which any person can start a node and start uploading and downloading files. It uses a Distributed Hash Table to address content using a combination of SHA-256 hashes and hyperlinks. This means there is a global key-value store for all files (and file parts), in contrast to BitTorrent, which works with torrent swarms and trackers. IPFS uses an efficient graph implementation (a Merkle DAG) for addressing content and nodes. In addition, it removes duplications of files across the network. As IPFS uses a global key-value store for all files, every file hosting node needs to synchronize with the latest acyclic graph before it can start serving files.

End-users of content stored on IPFS can access content without supporting the network, so there is the possibility of free-riding. For example, users of the aforementioned Audius music streaming system are not required to run an IPFS node to improve the health of the network. In addition, there are no direct (financial) incentives to run an IPFS node, other than to help the network and to host content.

3.5.3. Comparison: IPFS and BitTorrent

A notable difference between IPFS and BitTorrent is that IPFS makes a difference in file-hosting nodes and end-users (which only download files). BitTorrent does not make a distinction between these actors. In BitTorrent, every participant of the network has the capabilities to both upload and download content. Therefore a BitTorrent network using DHT is typically more decentralized than IPFS.

IPFS uses a global file index using a hash tree, which means that every two file that produce the same hash are stored on the same index. This leads to de-duplication, which may result in better use of disk space, in comparison to BitTorrent.

3.6. Incentives for file hosting

In a DAO, the party responsible for hosting and spreading of files is not well-defined. To tackle the tragedy of the commons, entities should be incentivised just enough for the system to be sustainable and usable, but no more. An example incentive system is bandwidth tokens (de Vos & Pouwelse, 2018) as part of the Tribler system.

Tribler (Pouwelse et al., 2008) is a peer-to-peer system to share, download and stream multimedia. It has implementations for desktop environments and an Android prototype. It makes use of BitTorrent for file transfer and adds anonymization techniques on top of it. In addition, it makes use of its bandwidth tokens: an incentive system to increase cooperation between users, in order to achieve high availability of downloads. In essence, it subtracts tokens for downloading content from peers and rewards tokens for helping peers. An overview of the Tribler desktop interface can be seen in fig. 3.2.

4
Design

In this section we present the design of our software application MusicDAO. MusicDAO is a mobile music streaming and discovery app, with peer-to-peer payment to artists in the form of donations and subscriptions. The MusicDAO is fully decentralized by design. This means there are no intermediaries, third parties or proprietary servers needed. All users of the app form a community to share audio tracks and transfer money. Any user can join this community, publish their musical works and receive money from its listeners. With the goal of distributing power in the music industry, every peer in the network will have the same rights and access to the same functions. All participants cooperate in the network, which makes it self-scaling by design.

The overall design of the system can be seen in fig. 4.1. This describes the interaction between the different components, libraries and frameworks. The following sections explain the designed features, components and design choices.

4.1. Serverless architecture

The network we are designing cannot use central servers or servers from third parties. This network should allow listeners and artists to exchange music and money, without the help from external services. We choose to build a phone-to-phone serverless network, for the following reasons. Firstly, Mobile phones have the largest share of any type of device using music streaming services. Secondly, if the software that we design and implement runs well on a network consisting of only mobile devices, we can conclude that it will run well on better hardware (such as PCs) as well.

We choose to use the Android framework for decentralized apps as proposed by Skala (2020). This gives us a toolbox for building phone-to-phone serverless apps and make use of its distributed ledger technology to build a public database. It also gives us a public-key infrastructure to identify and authenticate artists.

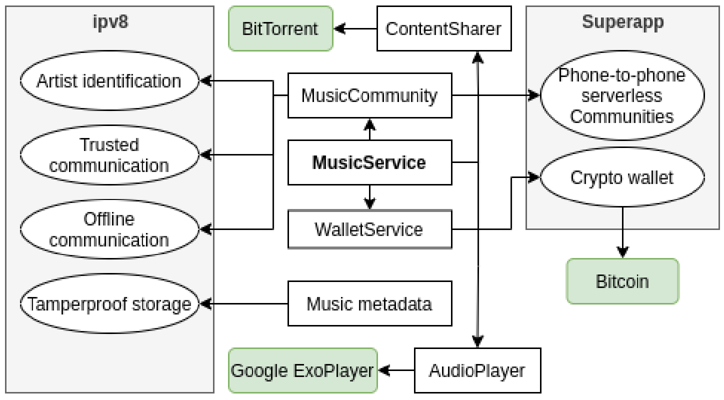


Figure 4.1: Architecture overview, with in green the external libraries. MusicService is the central component in our system

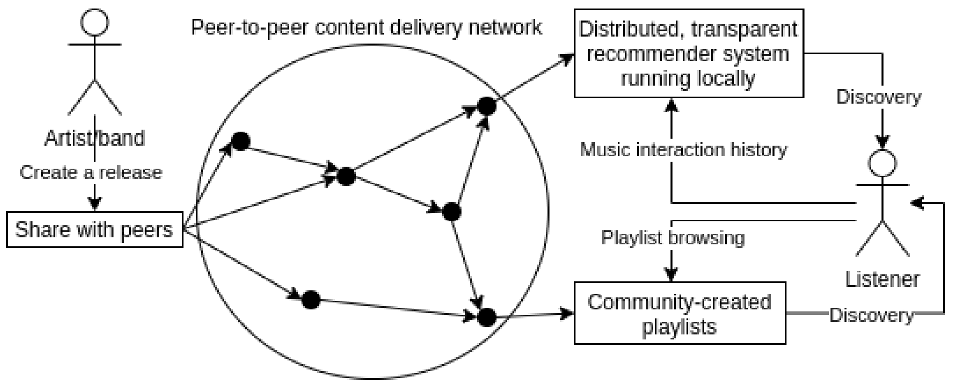

Artist/band
Create a release
Share with peers
Peer-to-peer content delivery network
Distributed, transparent recommender system running locally
Discovery
Music interaction history
Playlist browsing
Listener
Community-created playlists
Discovery

Figure 4.2: Desired music publishing flow using a distributed network

4.2. Open protocol and artist freedom

The design of our system contains both an open protocol and an application for the end-user. This is inspired by the ideas from Masnick (2019), who describes that the Internet should go back to open protocols instead of platforms, and that there should be a clear distinction between applications and protocols, so that users have the choice between different applications. Then, every application can have its own strategy of content moderation. This should result in more competition to “[...] provide better services that minimize the impact of those with malicious intent, without cutting off their ability to speak entirely” (Masnick, 2019).

In our context, we envision different applications using the same streaming, discovery and payment protocol, but with each application having strategies for content filtering and user interfaces. This way, music can not be censored in a centralized manner. Moderation of content happens on the side of the application, so that the user is in control of the settings of moderation and we do not lose freedom of speech or data resiliency.

4.3. End-to-end music delivery model

In contrary to the current music publishing situation, dominated by IT gatekeeping and oligarchs as visualized in fig. 2.1, we present the desired situation in fig. 4.2. This shows the liberation for artists in publishing their content, and the reduction of single-point-of-failure risks. In this system, artists are free in what they upload. In addition, their content can not be taken down by any authority unless there are no participants in the network. The discovery of content is done using open source, transparent systems and listening data is saved and processed locally.

To achieve this situation, a main component of our system is the storage of metadata and audio files for playlists. We design an abstract model for the structure of this metadata, so that the artist is free in the way to release music content. The artist may publish tracks as part of a single, an EP, an album or any other structured list of tracks, as a Release object.

A Release object contains a list of tracks that are published by a clearly identifiable artist or group of artists. It is modeled as shown in fig. 4.3. Release objects are shared between peers in the network. By discovering many of those objects, a user can see and browse through them to select a track to play. A Release object merely contains metadata of the tracks. We design the network to have a separate channel for downloading the track files. This is to enable fast discovery and searching of Releases, as Release objects have a small byte size.

4.4. Identification of participants

The MusicDAO allows any person to participate, and start publishing or listening to music. It requires a permissionless infrastructure, in which artists can be identified. As we design a system that is fully decentralized, we cannot use a central database to record user identities. Therefore every user generates a unique identity to be used in the network, and must be able to give proof of this identity. We use a public key infrastructure (PKI) which achieve these goals. Every user stores their private and public key on their device, and only share their public key. The keypair has a mathematical property that allows verification of messages that are signed with a private key. By comparing the public key of a peer with their signed message, anyone can verify the

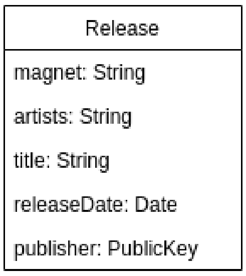


Figure 4.3: Release blocks structure as seen on TrustChain

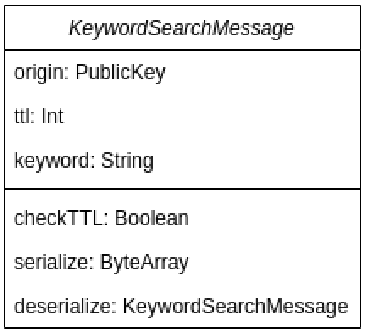


Figure 4.4: KeywordSearchMessage object sent over IPv8 in MusicCommunity

authenticity of the message.

In the context of MusicDAO we use this PKI to proof ownership of Release objects. All Release objects are signed using the owner's private key and the signature is added to the object. Any user receiving this object can verify its authenticity.

We choose to use the public key infrastructure as implemented in the TrustChain-Superapp (Skala, 2020). This gives us a unique identity per Android phone, by abstracting the network identities such as (changing) IP addresses and Bluetooth addresses.

4.5. Trust system and sybil-resilience

In any permissionless infrastructure, legitimacy of parties is not defined by a centralized authority. To still establish trust in the legitimacy of artists, we use the TrustChain DLT (Otte et al., 2017). Using this technology, we record the history of uploaded tracks in an immutable and transparent way. In essence, every artist adds Release blocks to its personal chain, and due to the interlinking mechanism of TrustChain, it is not possible to hide parts of this history. Every participant can view this timestamped history, as it is public by design.

This way, an application can inspect the legitimacy of an artist. For example, if an application finds a song X, published by both participant A and B, but the song published by participant A was published later, it can be concluded that A is more trustworthy than B, as B may have copied the song. This system can also be extended trivially to user ratings, or other interactions between parties, to achieve better measurements of trust. This distributed datastore of immutable and transparent histories then becomes a measure against Sybil attacks (Douceur, 2002) and artist impersonation.

4.6. Distributed storage

Central to our system is sharing downloading and storage of audio files and Release objects (see 4.3). To design a system which has no middlemen or regulators for publishing Releases, and has no central control, a distributed storage system is required. This storage system should have the following properties: immutability (data cannot be tampered with), resiliency (data should be available as long as users want it) and rigorous duplication (all objects should be saved on multiple machines). Distributed ledger technology (DLT) allows for these properties, so we design our system with a DLT as a major component.

One implementation of this technology is TrustChain (Otte et al., 2017) which allows for recording transactions between peers in a linearly scaling public ledger. Every peer has its own immutable and public blockchain which shows its history of transactions with others. This way we can establish trust in a certain party. In our context, we can use this mechanism to estimate trust in artists by inspecting their public history of uploads. We choose TrustChain because of its scalability, its trust mechanism, and because it has a native implementation for Android, as described in 3.3. In addition, this implementation allows for offline communication, so users can download and explore new content using Bluetooth or LAN.

4.7. Peer-to-peer music file sharing

To be able to have low latency for discovering and playing music tracks, while using no central nor high-throughput servers, the network demands participants to upload content continuously. We design the app to, by default, use the network capabilities of the mobile device to upload content as much as possible. This

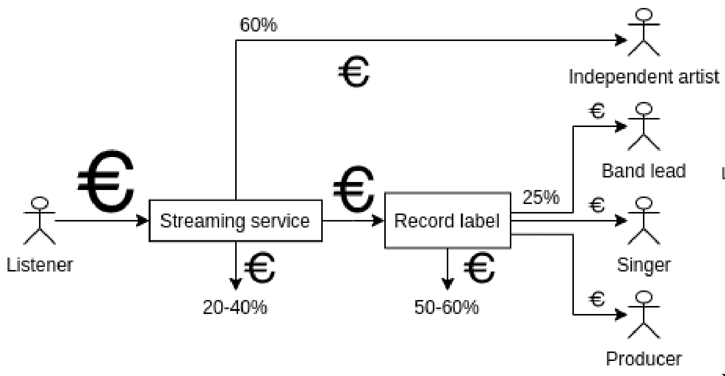


Figure 4.5: Money flow: current situation (simplified)

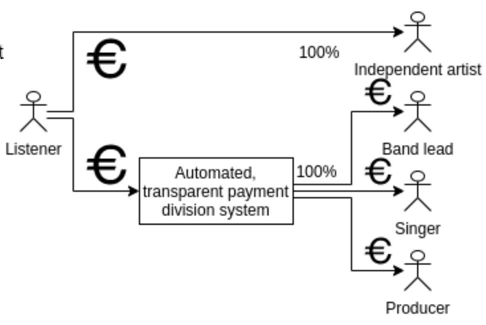


Figure 4.6: Money flow: desired situation

is constrained by networking hardware, data subscription plans and other software running on the phone.

The peer-to-peer file sharing protocol BitTorrent is suitable to share audio files in MusicDAO. We make BitTorrent a design choice as it does not require synchronizing with a global data store, in contrary to IPFS. This means we can build a metadata store independently from BitTorrent (for example, using TrustChain as explained in 4.6). BitTorrent also has stable implementations for Android.

4.8. Distributed search algorithm

For searching content we use introduce our simple distributed algorithm. Pseudocode of this algorithm is shown in algorithm 1. It asks peers around for content tagged with some keyword. When a peer finds a match on their local database, it sends this Release object to the original asking peer. Otherwise it forwards the query to their neighbours, after reducing the time-to-live property by 1. The messages stop being forwarded once their time-to-live property hits below 1. The structure of search messages are shown in fig. 4.4.

4.9. Transparent money flow

Figures 4.5 and 4.6 visualize, in a simplified fashion, the difference of how money flows in the current situation and in MusicDAO. It shows that, when intermediaries are cut from the flow, artists will have a higher income for the same fees from the listener. Streaming services and record holders introduce many overhead costs. Our system allows artists to publish their songs without the need to contact a label. The biggest difference in income will be seen for independent artists, as streaming services gives particularly low payouts for unsigned artists.

As we are designing a system with no intermediaries, it should be possible to give money directly to artists. Cryptocurrency allows for peer-to-peer payments which achieve this goal, so we use this in the MusicDAO. Cryptocurrency payments will be used for two different functionalities: a user can send a donation to an artist, or a user can pay artists using a monthly subscription system. This subscription system pays artists that the user listened to, using the Artist Income Division Algorithm (see 4.9.2).

In the desired money flow, we have a Fig. 4.6 shows another component: an automated and transparent payment division system. In practice, this should be an algorithm running locally on the machine of the user which calculates how much money should go to each of the shareholders of a particular song. Currently, record holders have this task, but there exists no transparent system for this, so they can give low payouts to its artists. We design the transparent payment system to be an immutable record on a distributed ledger, on which a specification of the exact shares per artists are written down. This can be implemented using TrustChain blocks (Otte et al., 2017).

We choose to use Bitcoin as a cryptocurrency as it has shown to be a secure and popular payment system, and it does not rely on any third parties to run. It also allows for making a experimentation environment without any high-throughput external servers.

4.9.1. Wallet

Cryptocurrency implementations allow for private/public key-pairs which can be interpreted as a kind of wallet; the funds can only be unlocked by a holder of the private key. In the case of MusicDAO we design the app to include a wallet for every user. To receive money, every artist should share their public key to all of their listeners. To achieve this, the public key of their cryptocurrency wallet is included as a property of the Release objects (see 4.3). As there are no institutions or banks involved in storing money, users will be required to keep their private key safe.

4.9.2. Artist Income Division Algorithm

To provide a stable income for artists, in the form of reoccurring payments, we design the Artist Income Division Algorithm. This algorithm calculates how subscription money is split into payments to artists. The user can enable a periodic payment. This money is then divided over the artists the user listens to, in proportion to the amount of interaction with each artist. Interaction can be measured in e.g. time listened, plays or feedback in the form of likes. The details of this division is explained in the implementation section of AIDA (see TODO).

Algorithm 1 Distributed algorithm for remote search

```
function DistributedSearch($query, ttl, maxPeers, minResults$)        ▷ Device initiates the search
    $results \leftarrow localSearch(query)$                            ▷ Filter local database
    if $|results| \leq minResults$ then
        $origin \leftarrow myAddress()$                                ▷ Address of the device initiating the search
        for $i \leftarrow 1, maxPeers$ do
            $peer \leftarrow peers[i]$                                 ▷ Select a random neighbor
            $sendQuery(origin, peer, query, ttl)$
        end for
    else
        return $results$
    end if
end function
function OnQuery($origin, peer, query, ttl$)                          ▷ This is called when a query is received
    if then$ttl \leq 0$
        return
    end if
    $ttl \leftarrow ttl - 1$
    $results \leftarrow localSearch(query)$
    if $|results| \leq 1$ then
        for $i \leftarrow 1, maxPeers$ do
            $peer \leftarrow peers[i]$
            $sendQuery(origin, peer, query, ttl)$
        end for
    else
        $sendResults(origin, results)$                                 ▷ Send the results back directly to the origin
    end if
end function
```

5 Implementation

The application is implemented as a 'mini-app' of the *TrustChain Superapp* (Skala, 2020). This follows the concept of super apps (Huang, 2019). The app is published on the Android Play Store[1] and runs on Android 5.1 and above. Its code is publicly available[2]. As programming language Kotlin is selected, as it is the preferred language for Android development (Haase, 2019). Moreover the underlying technology stack is also written in Kotlin, so this allows for neat integration. This section describes the implementation choices, usage of external libraries and presents the user interface of MusicDAO. The main interfaces of the app can be seen below (figs. 5.1, 5.2 and 5.6). An overview of the structure of the code can be seen in the package diagram in fig. 5.8.

5.1. Features overview

We implemented a peer-to-peer system to share, discover and listen to music, and to give payments to artists. 100% of money goes to artists. The system requires no servers, middlemen or third parties to run, and is resilient: it can not be taken down by any government or institution. Our Android app contains the following features:

- Defining album/single/EP releases using metadata, and sharing those with peers; Sharing of your audio tracks with peers; Immutable storage of music metadata and cryptographic identification of artists.
- Streaming music over BitTorrent; Prioritization algorithm to minimize streaming latency; Caching audio files and metadata.
- Browsing playlists; Local and remote keyword search for music.
- Mobile bitcoin wallet implementation; Peer-to-peer donations to artists using Bitcoin.

The following planned features were not implemented:

- Artist Income Division Algorithm.
- Content sorting algorithm based on swarm health.

Apart from the Android SDK and the TrustChain Superapp, the main libraries that are used are shown in 5.1. In the following sections we explain the details of implementing the aforementioned features in a top-down overview, starting with the interfaces.

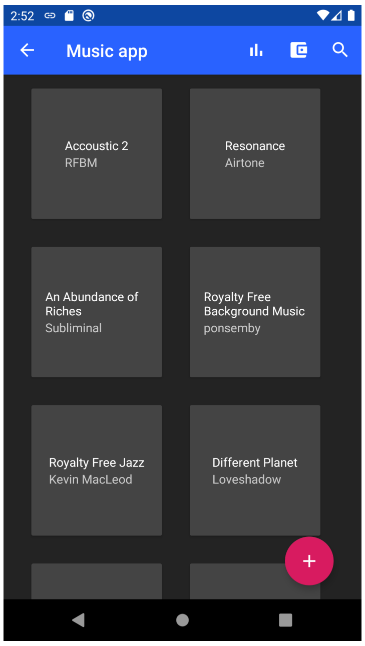
5.2. Playlist overview interface

Overall, the app is set in a dark-styled color scheme. The 'home' of the app is the playlist overview screen (see 5.1). It is the screen that is first shown upon starting the MusicDAO. Here the user is presented a list of playlists, with title and author, loaded from local disk and from peers. In our current implementation, each

[1] `https://play.google.com/store/apps/details?id=nl.tudelft.trustchain`
[2] `https://github.com/Tim-W/trustchain-superapp`

Name	Version	Usage
JLibtorrent	1.2.10	Peer-to-peer file distribution
TorrentStream-Android	2.7.0	Video streaming library
ExoPlayer	2.10.5	Android multimedia player
BitcoinJ	0.15.7	Java bitcoin interface
XChange	5.0.1	Crypto/USD price conversion

Table 5.1: Notable libraries used



Accoustic 2
RFBM

Resonance
Airtone

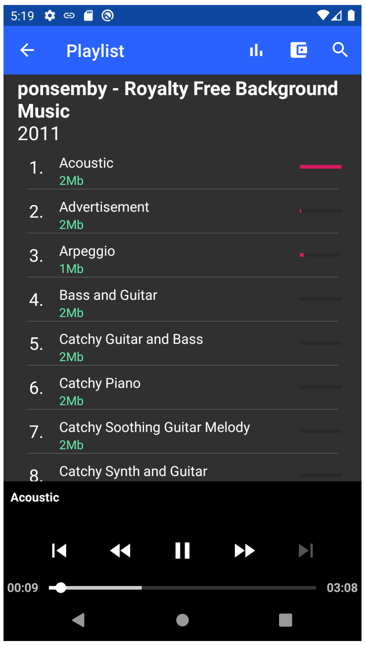
An Abundance of Riches
Subliminal

Royalty Free Background Music
ponsemby

Royalty Free Jazz
Kevin MacLeod

Different Planet
Loveshadow

Figure 5.1: The playlist overview screen, which is the entrance screen



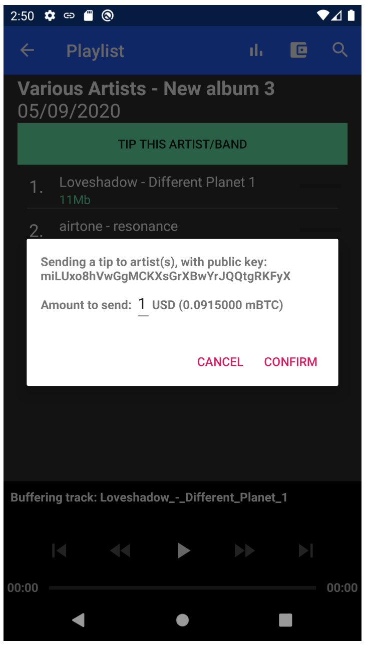
ponsemby - Royalty Free Background Music
2011

1. Acoustic
2Mb
2. Advertisement
2Mb
3. Arpeggio
1Mb
4. Bass and Guitar
2Mb
5. Catchy Guitar and Bass
2Mb
6. Catchy Piano
2Mb
7. Catchy Soothing Guitar Melody
2Mb
8. Catchy Synth and Guitar

Acoustic

00:09 03:08

Figure 5.2: Playlist fragment, showing all tracks of one Release



Various Artists - New album 3
05/09/2020

1. Loveshadow - Different Planet 1
11Mb
2. airtone - resonance

Sending a tip to artist(s), with public key:
miLUxo8hVwGgMCKXsGrXBwYrJQQtgRKFyX

Amount to send: 1 USD (0.0915000 mBTC)

Buffering track: Loveshadow_-_Different_Planet_1

00:00 00:00

Figure 5.3: Sending a tip to an artist or band

Playlist fragment corresponds to exactly one Release block (see 4.3). MusicDAO does not support user-made playlists as of the time of writing. The playlists are rendered in real-time based on TrustChain data. This means: during browsing, newly clickable playlists show up instantly once they are discovered from peers. The playlists are sorted on their torrent swarm health in ascending order. Caching is used for fast browsing and music playback. All torrent metadata and all track files received from peers are cached on the Android phone.

5.3. Playlist fragment interface

When the user selects a playlist to browse and play, this fragment is shown (see 5.2). Here, the user can select a track to play. It presents its list of tracks and other metadata, such as the title and artist(s). For each track the file size is displayed, and a loading indicator on the right side. This shows, in real time, how much of the track is downloaded. In the example of 5.2, the first track is fully loaded. The track player, shown on the bottom, interacts directly with the tracklist and the selected track. It shows which track is selected and whether it is currently playing or buffering.

5.4. Wallet interface

Each device participating in the MusicCommunity is given a private/public wallet identity upon installation of the MusicDAO. The wallet interface (fig. 5.4) shows synchronization status of the RegTest network (see 5.8). Once the wallet is fully synchronized with the blockchain, the private key and balance are displayed as shown in fig. 5.5.

Upon browsing a playlist, a donation button is displayed as shown in fig. 5.6. When pressing this button, the user can select an amount and make a direct donation to the artist or band, in the form of a bitcoin transaction from their wallet. The user enters a value in USD and the corresponding amount in bitcoin is calculated and shown when this field is edited. This is implemented using an external trading platform API[3][4]. After confirmation, the transaction is registered on the RegTest network (see 5.8).

As a side note, querying a trading platform API for currency rates makes the app reliant on an external server to run, which is not the aim of this thesis. However this function can trivially be removed without further implications on the system. The current system contains the function to improve the user experience.

5.5. Networking

We implement audio track uploading, downloading and streaming using JLibtorrent[5], an implementation of BitTorrent. Immutable and public storage of metadata is implemented with TrustChain (Otte et al., 2017).

5.5.1. Release creation and sharing

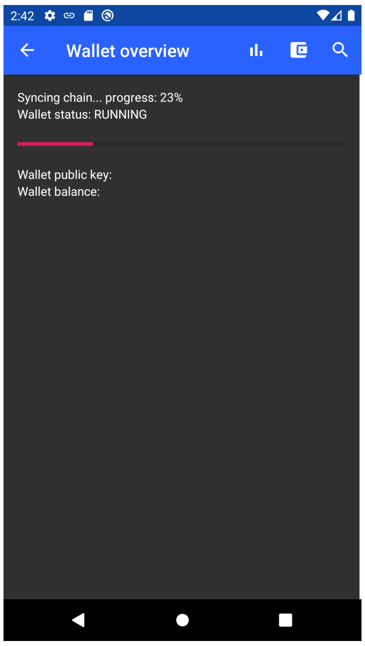
To create and share music content, a user can create a Release object using the dialog shown in fig. 5.7. There are two options for creation: the user either selects local audio tracks or pastes a magnet link. Afterwards, the user adds metadata describing the contents and submits the Release. In the background this creates a torrent file, which is stored on the mobile device. Finally, by using the computed infohash and file list, the magnet link of the torrent file is created and added to the *magnet* field of the Release block.

We use trackerless torrents, to keep the network decentralized. Instead of relying on centralized trackers, peers are found using a distributed hash table (*Distributed Hash Table*, 1981-2019). In addition, the app uses the local peer discovery (LPD) (8472, 2015) functionality of BitTorrent. This allows for finding peers and transmitting torrent pieces over a LAN, resulting in fast transmission and low latency.

5.5.2. Content seeding

Seeding of content is implemented using a simple continuous mechanism. The ContentSeeder class runs a background thread which seeds all local torrents, with an upper threshold T. This is set to $T = 10$. The ContentSeeder uses a last-in-first-out heuristic: Only the top T most recently created/received torrent files are seeded.

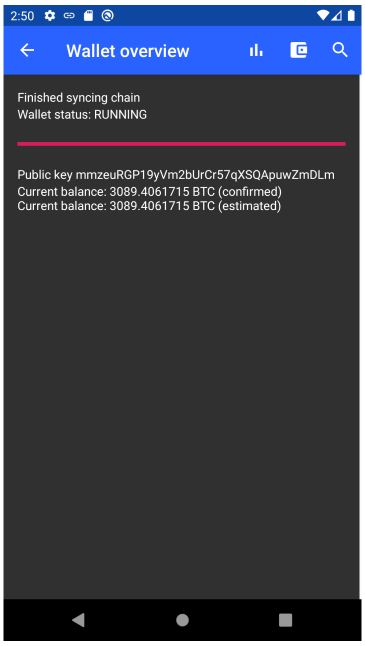
[3]`https://github.com/knowm/XChange/releases/tag/xchange-5.0.1`
[4]`https://www.binance.com/en/trade/BTC_USDT`
[5]`https://github.com/frostwire/frostwire-jlibtorrent`



Syncing chain... progress: 23%
Wallet status: RUNNING

Wallet public key:
Wallet balance:

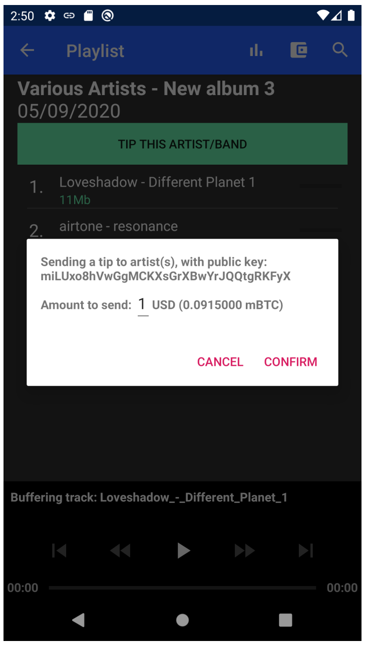
Figure 5.4: Synchronizing with the Bitcoin RegTest environment blockchain



Finished syncing chain
Wallet status: RUNNING

Public key mmzeuRGP19yVm2bUrCr57qXSQApuwZmDLm
Current balance: 3089.4061715 BTC (confirmed)
Current balance: 3089.4061715 BTC (estimated)

Figure 5.5: Wallet overview and balance after synchronizing



Various Artists - New album 3
05/09/2020

1. Loveshadow - Different Planet 1
11Mb
2. airtone - resonance

Sending a tip to artist(s), with public key:
miLUxo8hVwGgMCKXsGrXBwYrJQQtgRKFyX

Amount to send: 1 USD (0.0915000 mBTC)

Buffering track: Loveshadow_-_Different_Planet_1

Figure 5.6: Sending a tip to an artist or band

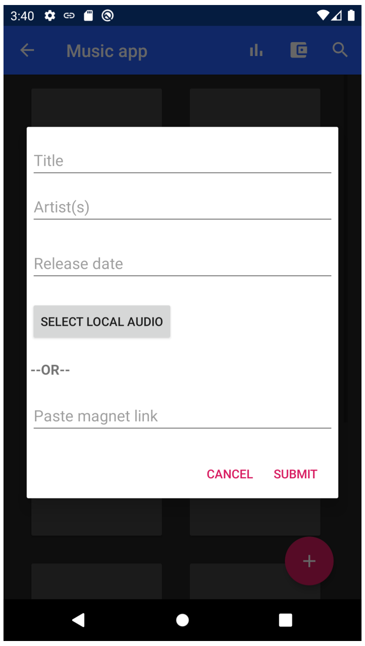


Figure 5.7: Dialog for creating and publishing a new Release

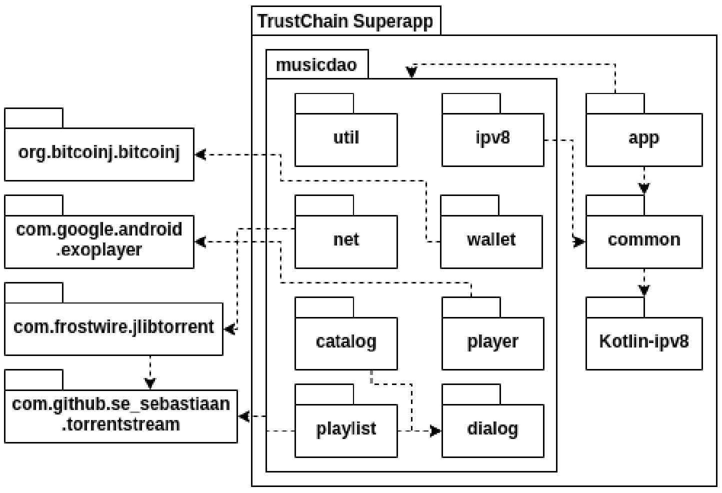


Figure 5.8: Package diagram showing the interaction with external libraries and TrustChain Superapp packages

5.6. Music Player and Streaming

Playing music is implemented using ExoPlayer (5.1). This music player library is suitable as it allows for playing tracks that are partially loaded, which enables streaming.

5.6.1. Priority handling

To achieve fast buffering, we implemented a priority system for tracks and for parts of tracks (chunks). In essence, the player prioritizes chunks that the user is currently interested in, by actively asking peers to send the corresponding chunks that are necessary to play the selected section of the track. In addition, the selected track is given a higher priority over other tracks in the playlist.

This uses the piece priorities system in libtorrent, which range from 1 (normal) to 7 (highest) (see libtorrent Manual[6]). By default, the first couple of chunks of each track are given a high priority, to reduce the chance of buffer underflow, so that the user can start streaming early.

5.7. Identity and authenticity

Every device participating in MusicDAO has a unique identity. MusicDAO implements the public-key infrastructure as described in 4.4 using the identity system proposed by Skala (2020). This uses the *Curve25519* cryptography system. Using this system, it is computationally infeasible to obtain the private key from a public key. For simplicity in implementation, we assume that every device in the network is an abstraction of a unique artist.

All immutable music release blocks are assigned an identity of the creator using a public key. Every device running the MusicDAO will receive a public/private key-pair upon first launch.

Currently there is no multi-signature support implemented. This means that in the case of a group publishing a Release, there is only one public key representing the whole group. Every artist, and every unique collaboration between artists should generate their own key-pair to describe ownership of the Release.

5.7.1. Keyword search

The app allows users to search for music content remotely using keyword search. The search function tries to find matches locally, and if there are only a few found, it will try to ask for content from peers. We implemented algorithm 1 for this functionality. The current implementation uses only a simple filter, which checks if the keyword is contained in the metadata of the music release.

When the user performs a search, the local database is filtered first to find matches. If there are unsatisfactory results, it sends a *KeywordSearchMessage* (see 4.4) to a few random peers. This asks neighbours to inspect their local database to find matches for the same query. If the peer finds a match, it sends the corresponding results directly back to the original asker. To disallow packages from endlessly being forwarded on the network we use a time-to-live property. For details, refer to algorithm 1.

5.8. Donations and payments

Our system contains peer-to-peer payments where 100% of money goes to artists. We created a public Bitcoin RegTest environment[7] to test peer-to-peer Bitcoin donations and payments. This creates a new 'clean slate' Bitcoin blockchain and allows for full control over the chain and miners. This enables a test environment that is useful for experiments, as we can tweak the block generation speed and keep track of all transactions registered on the blockchain.

Each device must be synchronized with the network in order to make payments. A background thread of MusicDAO establishes and maintains communication with bitcoin nodes, so that it does not interrupt the user experience. The progression of synchronization can be seen in the wallet interface. Synchronization with the RegTest network is done using a single bootstrap server, which is a hard-coded address in the app. This is necessary for running on a test net, as it will take an infeasible amount of time to guess the address of a running Bitcoin node, with only a few nodes. but in a real-world situation, bitcoin nodes should be found by querying peers for bitcoin node addresses.

In the current system, when a user makes a donation, this donation can only go to one artist. An automatic splitting system between different artists of one group or band has not been implemented yet.

[6] `https://www.libtorrent.org/manual-ref.html#file-format`

[7] `https://developer.bitcoin.org/examples/testing.html#regtest-mode`

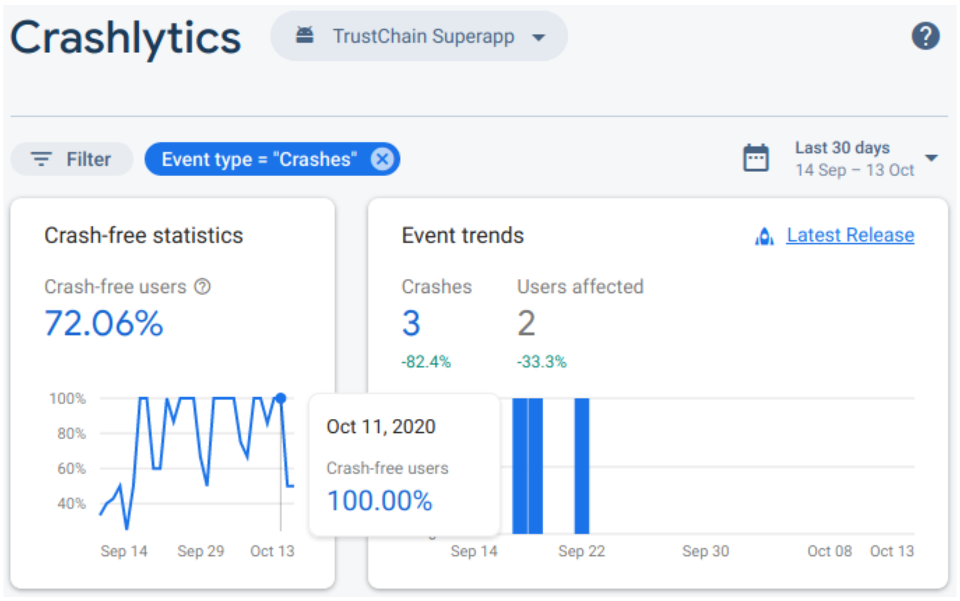


Figure 5.9: Firebase Crashlytics crash reporter; this figure shows the crashes over time

5.9. Quality assurance

In order to preserve quality of code we make use of continuous integration, automated tests and an online crash reporter. Unit tests are written using JUnit 4[8] and the mocking library Mockk[9]. The code coverage of these unit tests can be seen in 7.1.

These tests are run in an automated testing environment using Github Actions[10]. This continuous integration system is executed on every pull request for changes to the code. Aside from this, during the experimentation phase, app installs and crashes are recorded using Firebase Crashlytics, which is an end-to-end crash reporting tool. It shows details of each application crash of all users (see fig. 5.9). Using this, a full stack trace can be inspected remotely for each crash. In addition it helps with gaining insight in performance for certain types of Android devices, Android versions and helps us evaluate user interaction with the app.

[8]https://junit.org/junit4/
[9]https://mockk.io/
[10]https://github.com/features/actions

6
Release experiment

7 Evaluation

7.1. Code quality

For analyzing the code quality of the MusicDAO application, we can use code coverage measurements. The code coverage is shown in 7.1, measured by Android Studio 4[1]. The majority of uncovered code contain user interface interaction and networking callback logic. Code coverage could have been improved by introducing Android instrumented tests. These are tests that run on an Android device or emulator so that user interaction flows can be tested. However, we chose to not implement this as these type of tests can not be run in our continuous integration environment, and tweaking the CI for support of this would be a non-trivial task.

[1] `https://developer.android.com/studio`

Package/class	**Line coverage**
MusicService.kt	15% (15/95)
MusicBaseFragment.kt	50% (1/2)
catalog	24% (35/142)
dialog	20% (24/130)
ipv8	81% (57/70)
net	90% (27/30)
player	0% (0/50)
playlist	9% (19/203)
util	60% (42/70)
wallet	0% (0/100)

Table 7.1: Code coverage overview

8 Conclusion

Bibliography

8472, T. (2015). *Bittorrent local service discovery.* Retrieved September 14 2020, from `http://www.bittorrent.org/beps/bep_0014.html`

aCooke, C. (2018). *Dissecting the digital dollar.* London.

Aguiar, L., & Waldfogel, J. (2018). *Platforms, promotion, and product discovery: Evidence from spotify playlists* (Tech. Rep.). National Bureau of Economic Research.

Andersson Schwarz, J. (2016). Mastering one's domain: some key principles of platform capitalism.

Benet, J. (2014). Ipfs-content addressed, versioned, p2p file system. *arXiv preprint arXiv:1407.3561.*

Bucher, T. (2018). *If... then: Algorithmic power and politics.* Oxford University Press.

Buterin, V., et al. (2014). A next-generation smart contract and decentralized application platform. *white paper*, *3*(37).

Cohen, B. (2008). *The bittorrent protocol specification.* Retrieved September 19 2020, from `https://www.bittorrent.org/beps/bep_0003.html`

de Vos, M., & Pouwelse, J. (2018). A blockchain-based micro-economy of bandwidth tokens. *CompSys 2018.*

Distributed hash table. (1981-2019). Retrieved September 14 2020, from `https://encyclopedia2.thefreedictionary.com/Distributed+hash+table`

Douceur, J. R. (2002). The sybil attack. In *International workshop on peer-to-peer systems* (pp. 251–260).

Ellis-Petersen, H. (2014). Amazon and hachette end dispute on ebooks. *The Guardian.* Retrieved from `https://www.theguardian.com/books/2014/nov/13/amazon-hachette-end-dispute-ebooks`

Gillespie, T. (2014). The relevance of algorithms. *Media technologies: Essays on communication, materiality, and society*, *167*(2014), 167.

Haase, C. (2019). Google i/o 2019: Empowering developers to build the best experiences on android + play. Retrieved September 22 2020, from `https://android-developers.googleblog.com/2019/05/google-io-2019-empowering-developers-to-build-experiences-on-Android-Play.html`

Harvey, D. (2002). The art of rent: globalisation, monopoly and the commodification of culture. *Socialist register*, *38.*

Heap, I. (2017). Blockchain could help musicians make money again. *Harvard Business Review*, *5.*

Heuvelings, D. (2020). *Auxiety* (1st ed., Vol. 1). The address: Das Mag Uitgeverij B.V.

Huang, A. (2019). Super app or super disruption? Retrieved September 21 2020, from `https://home.kpmg/xx/en/home/insights/2019/06/super-app-or-super-disruption.html`

IFPI. (2018). *Global music report 2018.* International Federation of the Phonographic Industry London.

IFPI. (2020). *Ifpi global music report 2020 - the industry in 2019.* International Federation of the Phonographic Industry London.

Ingham, T. (2018). The odds of an artist becoming a "top tier" earner on spotify today? less than 1%. *Music Business Worldwide*, *25.*

Innis, H. A. (2007). *Empire and communications.* Rowman & Littlefield.

Jentzsch, C. (2016). Decentralized autonomous organization to automate governance. *White paper, November.*

Jia, B., Xu, C., Gotla, R., Peeters, S., Abouelnasr, R., & Mach, M. (2016). *Opus-decentralized music distribution using interplanetary file systems (ipfs) on the ethereum blockchain v0. 8.3.* Tech. Rep., 2016. Accessed: Jan. 2017.[Online]. Available: https://icosbull ....

Loewenstern, A., & Norberg, A. (2008). *Dht protocol.* Retrieved October 21 2020, from `http://www.bittorrent.org/beps//bep_0005.html`

Masnick, M. (2019). Protocols, not platforms..

Mulligan, M. (2020). Music subscriber market shares q1 2020. Retrieved from `https://www.midiaresearch.com/blog/music-subscriber-market-shares-q1-2020`

Otte, P., de Vos, M., & Pouwelse, J. (2017, 09). Trustchain: A sybil-resistant scalable blockchain. *Future Generation Computer Systems.* doi: 10.1016/j.future.2017.08.048

Pouwelse, J. A., Garbacki, P., Wang, J., Bakker, A., Yang, J., Iosup, A., ... Sips, H. J. (2008). Tribler: a social-based peer-to-peer system. *Concurrency and computation: Practice and experience*, *20*(2), 127–138.

Prey, R. (2020). Locating power in platformization: Music streaming playlists and curatorial power. *Social Media+ Society*, *6*(2), 2056305120933291.

ReCode. (2015). *Here's what happens to your $10 after you pay for a month of apple music.*

Rumburg, R., Sethi, S., & Nagaraj, H. (2018). *Audius: A decentralized protocol for audio content.*

Skala, M. (2020). *Technology stack for decentralized mobile services* (Unpublished master's thesis).

SNEP, E. . Y. (2015). *Bilan 2014 du marchÉde la musique enregistrÉe* (Tech. Rep.). Ernst Young.

Srnicek, N. (2017). *Platform capitalism.* John Wiley & Sons.