A Robot Economy for Music Without Any Intermediaries

Tim Wissel

TU Delft

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 Friday, December 18, 2020 at XXX.

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

TU Delft

Preface

Tim Wissel
Delft, October 2020

Contents

1 Introduction

Music streaming services have an immense amount of power in the music industry. The biggest streaming services dictate the rules which artists have to play by. The top 5 streaming services and the top 3 labels dominate the market. Artists have a hard time to make a living because the streaming companies take large revenue cuts. The centralization of power on Internet platforms cause unfairness in multiple industries, of which the music industry is particularly affected. This thesis aims to distribute the power from centralized streaming platforms to listeners and artists.

The music streaming industry is completely digital from recording and publishing to listening and digital rights management. We investigate the feasibility of fully replacing this digital value chain by software.

Insert clear Robot Economy definition, Insert contributions and goals of this thesis

1.1. Monopolization on the Internet

The Internet is moving from a network of people to a sparse selection of platforms over which nearly all commerce is regulated. A few Big Tech corporations are gaining increasing power in the the surface on which market exchange takes place. The consumer choice is diminishing due to the power of oligarchs and monopolies. As explained by (Stiglitz, 2019), there is a fundamental problem: the growing "concentration of market power, which allows dominant firms to exploit their customers and squeeze their employees, whose own bargaining power and legal protections are being weakened", while "[...] CEOs and executives are extracting higher pay for themselves".

1.2. Centralization of power in the music industry

An industry with great consequences of this trend is the music industry. In the last 20 years there has been a remarkably fast shift from the exchange of CDs in various stores to music streaming on the Internet. Music platforms and labels use their economic muscle to push down artist salaries. They take large cuts of revenue from the user subscription money.

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 a 25% to 40% revenue cut.

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 companies 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.

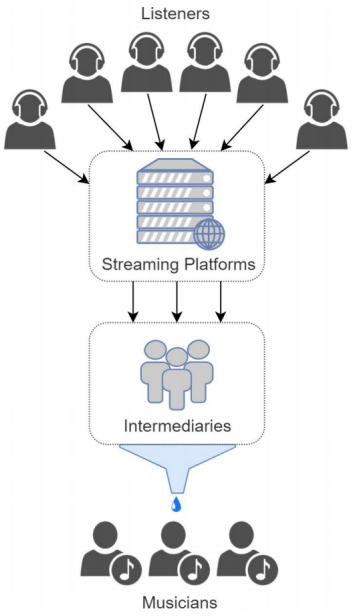

Figure 1.1: Artist compensation inconsistency

1.3. 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. A DAO is defined by Buterin (2014) as an "entity that lives on the internet and exists autonomously, but also heavily relies on hiring individuals to perform certain tasks that the automaton itself cannot do" (see 3.1). In this thesis we present the design and implementation of our mobile android app MusicDAO: a music streaming service for the common good. Users of this app form a phone-to-phone zero-server network over which they publish music, download music and transfer money. This proof-of-principle of a DAO shows a fair and transparent music streaming service in which no external servers, third parties or intermediaries are necessary. 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.

1.3.1. 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 biggest music streaming services run centralized, proprietary and closed-source software. The top 5 streaming services have a combined market share of 82% (see 2.1), and have huge power over both the producer and receiver sides of music streaming. 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 a black box to the user. This gives streaming companies curatorial power. A visualization of the economic muscle of both the label oligarchy and streaming platform oligarchy is shown in 2.1.

At the time of writing, there exists no alternative decentralized and transparent music streaming system with peer-to-peer payments directly to artists.

2.1. History in the industry: centralization of power

The nature of distributing music has changed spectacularly in the last 30 years. There has been a remarkable shift from physical sales, to online track downloads and piracy issues, to digital on-demand streaming. The bargaining power in music sales was once distributed over many different physical stores and labels, but is nowadays in the hand of a few labels and Big Tech corporations. The core problem follows: "A large number of sellers (musicians, singers, bands) are in interaction with a very small numbers of buyers" (Rayna & Striukova, 2009).

In the CD era of music, every city would have one or a few stores selling physical records. There were many music distribution companies competing for their sales.

With the rapid shift towards digital sales around the 2000s, IT companies used their advantage in terms of network infrastructure to sell digital copies to massive audiences. The downfall of the physical record stores began. At the same time, only a dozen digital stores managed to attract large audiences on their platforms, and thus survived. This marked the start of *platform accumulation* (Meier & Manzerolle, 2019): routing all music and money flow over centralized platforms.

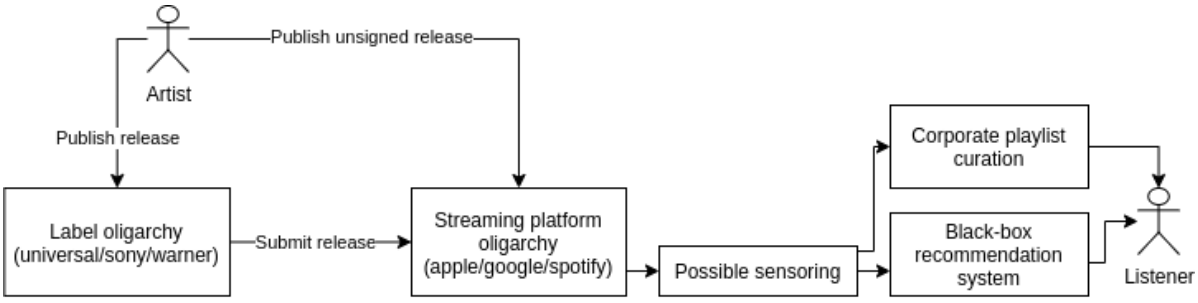

Figure 2.1: The current flow from publishing music to receiving it as a listener. The figure shows that the framework on which music is published and found is dominated by label and streaming oligarchs and by corporate decisions.

Streaming service	Market share
Spotify	32%
Apple Music	18%
Amazon Music	14%
Tencent Music	11%
YouTube Music	6%
Total	**82%**

Table 2.1: Global music streaming market share, measured by subscriber share (MidiaResearch, 2020)

Around the 2010s, the attention started to turn again to a new industry: on-demand streaming of music. In 2017, streaming accounted for 62% of music revenue in the US, compared to only 15% in 2012[1]. Nowadays, the number of distributors to choose from is reduced to only 5 Big Tech platforms. Together they form over 80% share of all subscribers. Clearly, streaming royalties are an increasingly important piece of income for musicians. However, these increased revenues are not felt in the pockets of musicians, but rather in major label and platform profits.

This history shows a trend towards centralization of power in the music industry. The future of artists is in the hand of a few large corporations, as they have monopoly power over them. This comes with several issues, as explained in the next section.

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 2.2). The top 5 streaming services, controlling 80% market share (MidiaResearch, 2020), have the power to ask high subscription fees. The Big Tech corporations behind these services are also tightly intertwined with dominant labels. For instance, Aguiar and Waldfogel (2018) notices that “major record labels have substantial ownership stakes in Spotify”.

According to IFPI (2020), 2019 became the first year in which digital streaming is the single biggest source of revenue for the music industry globally. 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[2] 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. 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.

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).

[1] `http://www.riaa.com`
[2] `https://www.bloomberg.com/opinion/articles/2017-09-25/the-music-business-is-more-unfair-than-ever`

	Music release	Label cut	Platform cut	Artist/band cut	Streams per month to earn min. wage (solo musician)
TIDAL	Unsigned	0%	40%	60%	117,760
	Signed	55%	50%	20%	353,280
Spotify	Unsigned	0%	40%	60%	287,574
	Signed	55%	25%	20%	862,722
Apple Music	Unsigned	0%	40%	60%	200,272
	Signed	55%	25%	20%	600,816
Google Play	Unsigned	0%	40%	60%	217,752
	Signed	55%	25%	20%	653,256
This thesis challenge	Unsigned	0%	0%	100%	<75,000

Table 2.2: Overview of revenue cuts (estimated) on streaming platforms, with a note on the streams/plays per month that an artist should have in order to make a minimum wage. The challenge of this thesis is to liberate artists from depending on intermediaries that take a large revenue cut. Sources: Trichordist (2014), DigitalMusicNews (2018).

2.4. Censorship issues

In relation to gatekeeping, platforms are now given the task to perform moral judgments on content, for example whether to censor a certain artist. This is controversial as these judgments are no longer in the hands of democracies but rather in the hands of companies. Recent issues exist such as the disappearance of Li Zhi[3], 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[4]

2.5. Slow and inaccurate royalty payments

An additional problem of the current complex flow of intermediaries result in slow and inaccurate royalty payments. As reported by BBC (2019), Eminem's publisher sued spotify because he has never been properly paid for songs that are streamed on Spotify for billions of times. The court mentioned that the streaming service "[...] lacked the infrastructure needed to make sure songs are licensed and musicians are paid". Royalty payments can take 6 to 12 months to arrive at artists (TODO cite/expand).

2.6. 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 its in-house curated playlists "puts Spotify in control of the demand curve" [5].

2.7. 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

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

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

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

choosing. This shows “curatorial power”: the ability to advance own interests by organizing and prioritizing content (Prey, 2020).

Musicians and record labels are increasingly 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.

A notable problem with this situation is that 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 the artist lacks knowledge in the algorithm workings. Frustrating for artists and labels is also the vagueness of getting playlisted: it is unclear why “[...] a particular track was placed, or replaced, on a playlist” (Prey, 2020).

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 depicted 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.

2.8. Research question

All in all, the main issue is as follows. The music industry is imbalanced: it suffers from centralized power that is in the hand of a few labels and Big Tech corporations. Our research question thus follows:

How can we design and implement a music streaming service that distributes the power from one authority to its users?

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 Big Tech, and more specifically 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.

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 technologies and organizational theories which enable autonomous communities to solve these problems by organizing themselves.

3.1. Decentralized Autonomous Organizations

As an alternative organizational framework for distributing money and music, we examine a recent concept and technology: Decentralized Autonomous Organizations (DAO). A DAO is "an entity that lives on the internet and exists autonomously, but also heavily relies on hiring individuals to perform certain tasks that the automaton itself cannot do" (Buterin, 2014). It is not owned by a single person or legal entity. It should also not require a specifically specified party to operate. For example, it should not depend on a single server or database, but rather have flexibility in adopting resources. As such we say it lives *autonomously*. Buterin (2014) also notes that a DAO should have *internal capital*: 'some kind of internal property that is valuable in some way, and it has the ability to use that property as a mechanism for rewarding certain activities'. A DAO is non-profit by nature, as there is no legal owner of the system.

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 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".

3.2. AI DAO

Central to this thesis is the real-world experimentation of an intelligent decentralized system: a system that is on the boundary of DAO and AI technologies (see 3.1). This means humans are still involved with performing some tasks and governing, but an intelligent agent can perform other decision making tasks. This may be the holy grail in automation. In our context, we envision human tasks to be: creating music and giving feedback on music (through e.g. donations). 'Robot tasks' could be e.g. (1) processing an automatic subscription payment system, in which all user money is divided fairly over the artists every time iteration; Or (2) tracking connectivity stats of other users; Or (3) determining trust of authenticity and identity of artists.

Much is still unknown in this research area. This thesis aims to create more knowledge on a 'AI DAO' by attempting to build a proof-of-concept of such intelligent system.

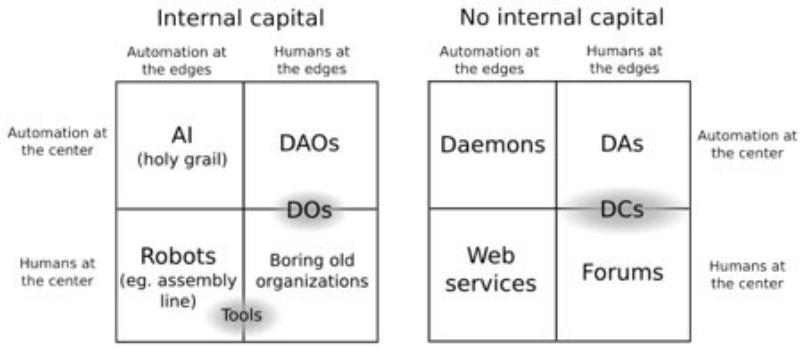

Figure 3.1: Decentralized Autonomous Organization, in comparison to other organizational structures. In a DAO, activity is performed by humans at the edges, automation at the center. It is an interplay in which robots and humans perform tasks. Source: Buterin (2014)

3.3. 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).

3.3.1. 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.

3.3.2. 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 3.6). 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 3.4). Using cryptocurrency, users spend their funds on these smart contracts, to unlock access to audio files.

3.3.3. 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 3.4). 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. 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.

3.3.4. Comparison

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 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.

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

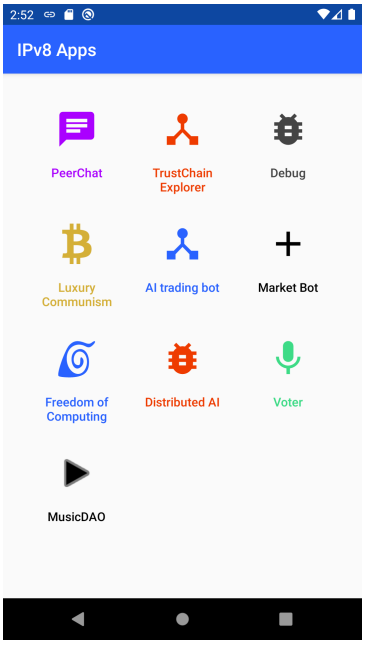

Figure 3.2: Trustchain-Superapp overview

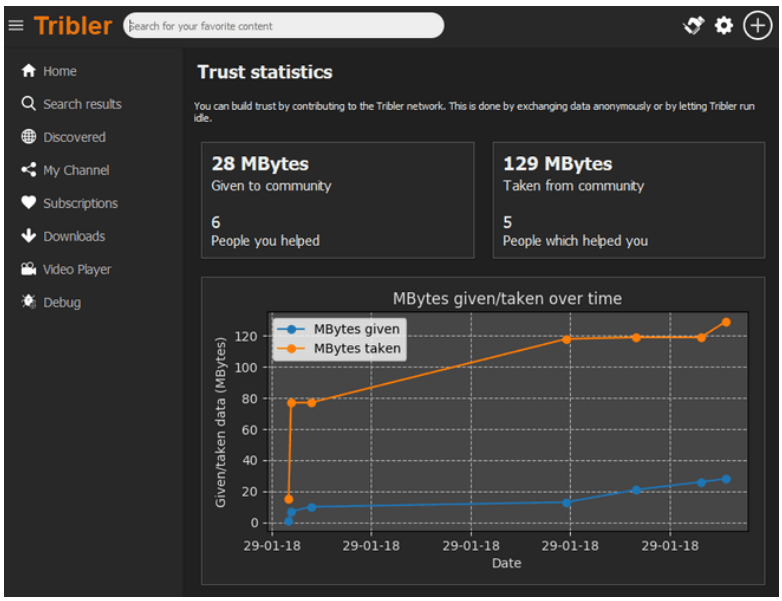

Figure 3.3: Tribler desktop interface, showing the bandwidth incentive system overview

3.4. 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.

3.5. 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. 3.2. Its mini-apps implement decentralized democratic voting and has bitcoin payment integration, among other features.

In the context of DAO, the TrustChain-Superapp contains a mini-app for decentralized governance using voting (see fig. 3.4). 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.

3.6. 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 technologies are BitTorrent and IPFS.

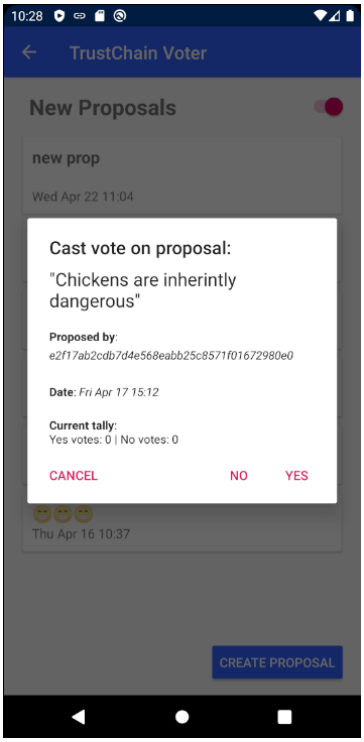

Figure 3.4: Distributed democratic voting mini-app, showing proposals

3.6.1. BitTorrent

BitTorrent (Cohen, 2002) is an open peer-to-peer file sharing protocol. It invented by Bram Cohen, and has generated a massive influence on network traffic on the Internet since its release. Today, it is still the most popular peer-to-peer protocol for sharing data. In 2019, BitTorrent was measured to generate 2.5% of all download and 24.6% of all upload bandwidth (Marozzo, Talia, & Trunfio, 2020). It went through many iterations and improvements. It has a large community, with over 3K repositories on GitHub related to the technology. Furthermore there are a few companies maintaining clients (such as `https://bittorrent.com` and `https://www.utorrent.com`).

In essence, BitTorrent makes use of seeding while downloading: this means that when multiple clients download a file, they simultaneously share pieces of this file with the other downloaders. This results, with well connected and performing peers, in fast downloads.

There are no differences between hosters and downloaders. 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 proposed solutions for this, discussed in 3.7.

3.6.2. 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. Protocol Labs[2], the team behind this technology, was founded in 2014 by Juan Benet. As of 2020, the team has grown globally to members from 19 countries, with substantial investments. However, there has not been large-scale adoption of this technology yet.

At its core, IPFS maintains a global key-value store for all files (and file parts). This is in contrast to BitTorrent, which works with torrent swarms and trackers. IPFS uses an efficient content addressing mechanism. Another feature is built-in file de-duplication. Additionally it supports public file history bookkeeping.

End-users of content stored on IPFS can access content without supporting the network, so there is the possibility of free-riding. In addition, there are no direct (financial) incentives to run an IPFS node, other than to help the network and to host content.

[2] `https://protocol.ai/`

3.6.3. 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.

3.7. 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 (J. A. 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. 3.3.

4

Design

In this section we present the design of our software application MusicDAO. MusicDAO is a mobile music streaming and discovery application, 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. It is a first step towards a fully autonomous and zero-cost music streaming industry. It is non-profit by design, as there is no single leader or company controlling it. Instead, its users determine its future. All users of the app form a community to share audio tracks and transfer money using mobile devices. Any user can join this community, publish their musical works and receive money from its listeners. 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.

Main goal of our system

The main goal of MusicDAO is: Distributing the power in the music industry, from centralized platforms to listeners and artists. Meaning: liberating artists from their dependence on money-grabbing platforms, so that they receive 100% of the subscription/donation money from their listeners.

4.1. Zero-cost autonomous music industry

We design a system that takes important first steps towards a zero-cost autonomous music streaming industry, with no intermediaries. In this utopia, intermediaries that add no real value to the industry receive no money. Artists receive near-100% of all income as they are the core contributors to the industry. Anyone

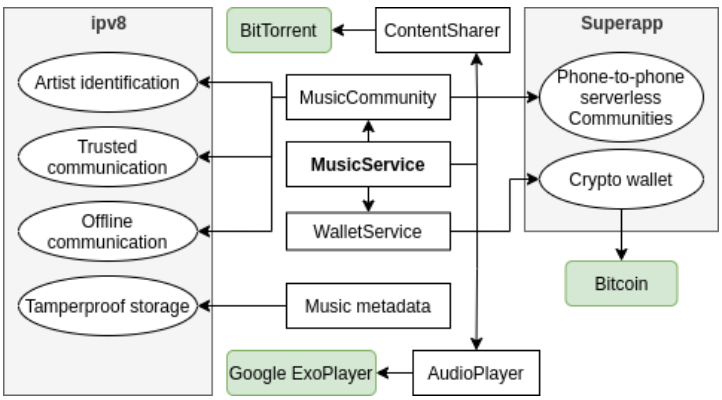

Figure 4.1: Architecture overview, with in green the external libraries. MusicService is the central component in our system

in the community of artists and listeners can create and share music without contacting a party for a contract or allowance. An open protocol over which money and music is exchanged, can be used with different applications, so that users have a choice of user interface.

Real-world thriving examples such as Linux or Wikipedia are driven by community and effort instead of profit. Consensus is reached through discussion instead of through pyramid schemes. We envision a similar transition for the digital music industry. The next sections will explain the design of our proof-of-concept of MusicDAO, which aims to be the first piece for reaching a fair music industry.

4.2. Phone-to-phone censorship-free network

We design a network which only consists of mobile phones. Every phone cooperates by storing, sharing and validating content. Each mobile phone has access to the same functionalities. The reason for this topology is to have no single, powerful party or centralized server, and that scales naturally. With the latter is meant: the computational power and bandwidth capacity grows with the amount of devices requiring these resources.

A proof-of-concept will show an important step towards mobile infrastructure for the common good: a system in which participants do not lose money and power to greedy intermediaries. Instead, they will benefit from cooperation. This concept aims for absolute fairness, controlled by the community from the ground up instead of dictated top-down. All money going into the system is divided over the participants, following rules written in code that are defined by the community.

4.2.1. Censorship-resiliency

A key attribute of the network is censorship-resiliency. That means that no single authority (company, institution or government) can remove tracks, or prevent devices from participating in the network. Censorship-resiliency is an important requirement for the system because of the issues described in 2.4. Attempting to build a resilient system while using Internet technologies results in a few key challenges as identified by J. Pouwelse (2012) and et al. (2014). To articulate these challenges, we specify a powerful adversary which has the goal to reduce the freedom of a user of the system. The adversary is known to manage the following attacks:

- Eavesdropping.
- Killing Internet switches.
- Direct censorship: installing malware or spyware.
- IP Filtering: block or filter content by restricting access of a specific IP address.
- Content filtering: removing or hiding specific content from several hosts.
- Confiscating devices: manually take some devices down (a small proportion of the whole network).

The following paragraphs explain how MusicDAO is designed to defend against these attacks.

Eavesdropping: In a censorship-resistant network, devices talking to each other must trust the network infrastructure to be free of eavesdropping. Preventing eavesdropping can be achieved using end-to-end encryption. However, an adversary may take down certain networking infrastructure to prevent communication altogether (Killing Internet switches).

Killing Internet switches: An alternative approach is to establish *direct communication* between devices, which means that data is transmitted device-to-device, without the use of a central access point such such as a router. We use BlueTooth LTE (Townsend, Cufí, Davidson, et al., 2014) as a back-up strategy for when Internet traffic is being eavesdropped or when Wi-Fi is killed. This is a widely supported technology in present-day mobile phones and has existing integration into the framework by Skala (2020).

Direct censorship is another challenge in designing an application. Direct censorship means that there is a piece of code installed on-device that tracks user activity, without the user knowing this. Our solution is two-fold. Firstly, the ecosystem is by default open source. Secondly, the evolution of the code-base is determined by a governance system and majority voting. This means that proposals to change the code-base will need to be voted on by its community. This idea is inspired by previous work from Jentzsch (2016).

Content filtering is not a threat to MusicDAO, because it uses an immutable data structure (see 4.8 for details). This means, in current context, that it is not possible to change or remove any (meta)data of music tracks after they have been published. In addition, multiple copies of track files are available through content

duplication. The system uses simple duplication heuristics, such that all objects that a phone receives is stored in cache. This strategy also defends against *confiscating devices* and *IP filtering*. When an adversary takes down, or blocks, a small portion of devices from the network, there will still be back-ups available on the other devices.

4.3. Phone-to-phone connectivity

As mentioned above, MusicDAO is designed to have a network consisting of only mobile phones. Key challenges to establishing and maintaining such a network are: discovering other devices, network connectivity, longevity and scalability.

Every device that wants to participate, will try to find other devices to join the network. Every device also keeps track of a routing table, containing public IP addresses of connected and connectable peers, including latency for each peer. This routing table is inspired by the routing table implemented by BitTorrent DHT (Loewenstern & Norberg, 2008). To discover an *initial* list of devices, we use a bootstrap server, which keeps track of other devices on the network to connect with. A bootstrap server should not be necessary when there are devices on the same Local Area Network that can introduce a new device to the network. However, the network of MusicDAO will be sparse at the start. After a few devices are discovered, the bootstrap server can be disregarded, as new devices are then trivially found via neighbor search.

As there will be no central server, each device acts as both a client and server. As devices may be behind routers or other *Network Address Translators* (NATs), establishing a direct connection between devices is not trivial. To achieve this, we use network address translation (NAT) traversal. NAT traversal is a set of methods used to establish a connection between devices which have no static, public IP address.

To support a healthy evolution of the network, every device will maintain a list of connected devices. Devices will send periodic *keepalive* messages to its connected peers to track which are alive and reachable. By doing this, devices can decide which peers are healthy connections. This list should grow to a few dozen, so that there are always connectable peers.

4.4. 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.

4.5. 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

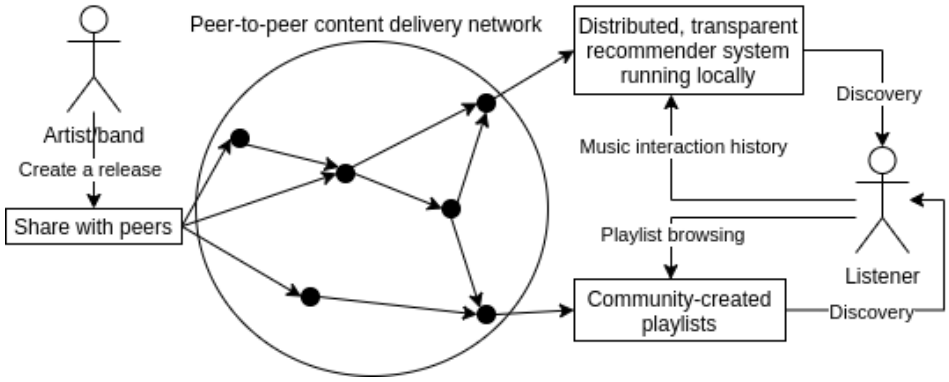

Figure 4.2: Desired music publishing flow using a distributed network

Release
magnet: String
artists: String
title: String
releaseDate: Date
publisher: PublicKey

Figure 4.3: Release blocks structure as seen on TrustChain

KeywordSearchMessage
origin: PublicKey
ttl: Int
keyword: String
checkTTL: Boolean
serialize: ByteArray
deserialize: KeywordSearchMessage

Figure 4.4: KeywordSearchMessage object sent over IPv8 in MusicCommunity

the track files. This is to enable fast discovery and searching of Releases, as Release objects have a small byte size.

4.6. 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 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 abstracts network identifiers such as IP addresses, which may change over time; so it provides a unique identity per Android phone.

4.7. Establishing trust and reducing sybil attacks

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.

4.8. Distributed storage

Central to our system is sharing downloading and storage of audio files and Release objects (see 4.5). 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 3.5. In addition, this implementation allows for offline communication, so users can download and explore new content using Bluetooth or LAN.

4.9. Peer-to-peer music 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 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 4.8). BitTorrent also has stable implementations for Android.

4.10. Distributed search

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.

4.11. 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 4.11.2).

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

Algorithm 1 Distributed algorithm for remote search

```
function DISTRIBUTEDSEARCH(query, ttl, maxPeers, minResults)        ▷ Device initiates the search
    results ← localSearch(query)                                    ▷ Filter local database
    if |results| ≥ minResults then
        return results
    end if
    origin ← myAddress()                                            ▷ Address of the device initiating the search
    for i ← 1, maxPeers do
        peer ← peers[i]                                             ▷ Select a random neighbor
        sendQuery(origin, peer, query, ttl)
    end for
end function
function ONQUERY(origin, peer, query, ttl)                          ▷ This is called when a query is received
    if ttl ≤ 0 then
        return
    end if
    ttl ← ttl − 1
    results ← localSearch(query)
    if |results| ≤ 1 then
        return sendResults(origin, results)                         ▷ Send the results back directly to the origin
    end if
    for i ← 1, maxPeers do
        peer ← peers[i]
        sendQuery(origin, peer, query, ttl)
    end for
end function
```

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 fully peer-to-peer, 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.

4.11.1. 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 4.5). As there are no institutions or banks involved in storing money, users will be

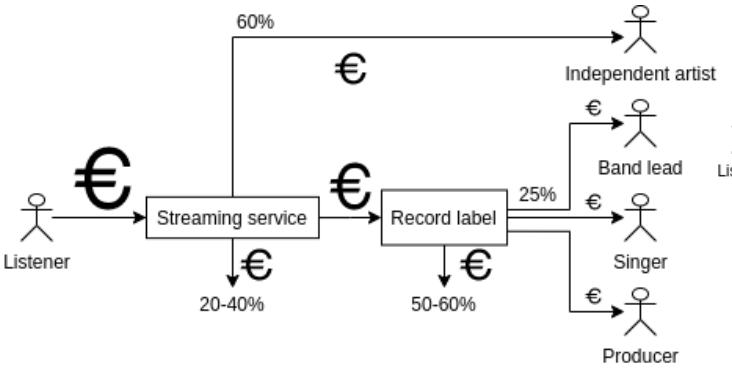

Figure 4.5: Money flow: current situation (simplified)

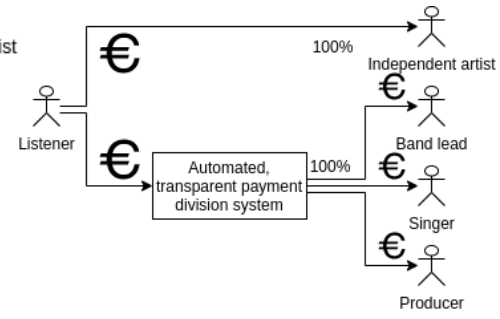

Figure 4.6: Money flow: desired situation

required to keep their private key safe.

4.11.2. 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).

4.12. Content popularity gossip protocol

4.13. Scalability

MusicDAO is scalable by using a scalable accounting system (see 4.8) and distributed file system (see 4.9). Every device records its own history. When a device wants to publish music, it only transacts with one other party, to sign the metadata record. As such, there is no global consensus, nor a global ordering of transactions, so all records are processed in parallel.

In addition, by using BitTorrent as the streaming/content layer, any participant can publish a torrent with music, without needing to interact with any other peer.

Finally, the transaction system, Bitcoin, may become a bottleneck in scalability. It requires global consensus and a voting process for every transaction. However, as of the time of writing, there is no mature and more scalable alternative as peer-to-peer payment system technology. We do not consider centralized technologies as that would concentrate power, in contrary to the aim of this thesis.

5

Implementation

We implemented our system on a network of phones, with no servers. Every mobile phone cooperates by sharing music and establishing trust. This means that the computational capacity grows with the amount of users, as does the demand on such capacity. The system is built along the ideologies of a DAO: there is no single party, server or database that the system requires to operate. As such, it is a non-profit system, where the artists keep all revenues. In MusicDAO, mobile phones are able to transfer money and music. In this trust-less system, trust in other parties is established through proofs of identity using cryptography, and publicly verified append-only history data. It is resilient against censorship-attacks: it cannot be taken down by any government or institution.

5.1. Zero-server infrastructure

The system described in the Design chapter is implemented on a network of phones, with no servers. This results in a fully distributed, leaderless organization. Every mobile phone cooperates by sharing music, transacting money and establishing trust. This means that the computational capacity grows with the amount of users, as does the demand on such capacity. The system is built along the ideologies of a DAO: there is no single party, server or database that the system requires to operate. As such, it is a non-profit system, where the destination of money flow is decided by its users, instead of by a single organization. In MusicDAO, mobile phones are able to transfer money and music. In this trust-less system, trust in other parties is established through proofs of identity using cryptography, and publicly verified using append-only history data.

5.2. Phone-to-phone Android application

MusicDAO 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[3]. 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.

Features overview

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.

[1] `https://play.google.com/store/apps/details?id=nl.tudelft.trustchain`

[2] `https://github.com/Tim-W/trustchain-superapp`

[3] `https://android-developers.googleblog.com/2019/05/google-io-2019-empowering-developers-to-build-experiences-on-Android-Play.html`

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

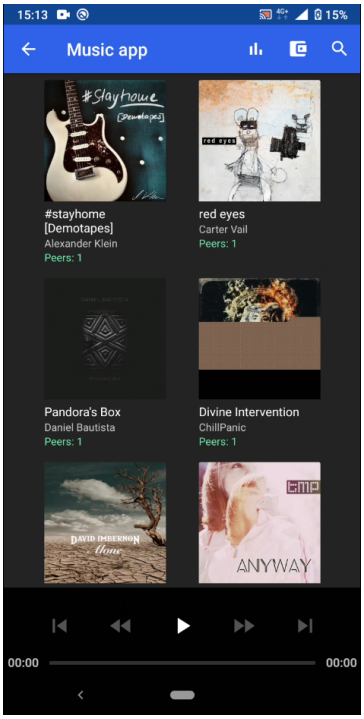

Figure 5.1: The playlist overview screen, which is the entrance screen

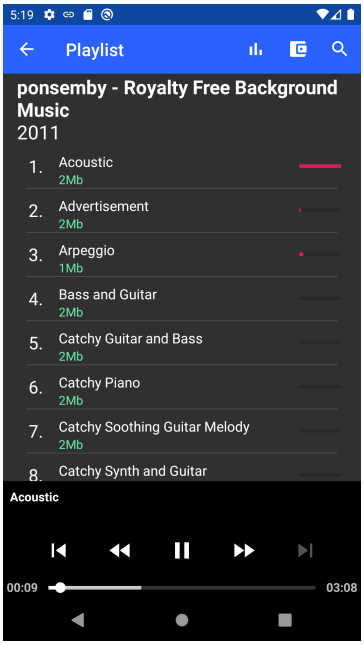

Figure 5.2: Playlist fragment, showing all tracks of one Release

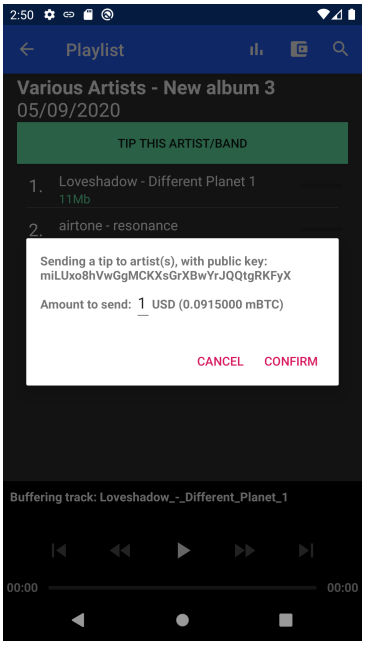

Figure 5.3: Sending a tip to an artist or band

- Browsing playlists; Local and remote keyword search for music.
- Mobile bitcoin wallet implementation; Peer-to-peer donations to artists using Bitcoin.
- Content sorting algorithm based on swarm health.

The following was not implemented:

- Artist Income Division Algorithm.

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.

5.3. Peer-to-peer content discovery

Peer-to-peer content metadata discovery is implemented by a push-based content spreading mechanism. All devices have an internal clock. Every i seconds, they send a random block of music metadata to a random peer. We analyze the impact of i on content discovery in chap. 6.

The entry point 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 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.

5.4. Music streaming mechanism

To achieve fast buffering of music, 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[4]). 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.

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. When the user selects a playlist to browse and play, the fragment in fig. 5.2 is displayed. 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.

5.5. Peer-to-peer financial infrastructure

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 5.9). 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[5][6]. After confirmation, the transaction is registered on the RegTest network (see 5.9).

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.

5.6. Networking

We implement audio track uploading, downloading and streaming using JLibtorrent[7], an implementation of BitTorrent. Immutable and public storage of metadata is implemented with TrustChain (Otte et al., 2017).

5.6.1. Release creation and sharing

Peer-to-peer content discovery is implemented by a push-based content spreading mechanism. All devices have an internal clock. Every i seconds, they send a random block of music metadata to a random peer. We analyze the impact of i on content discovery in chap. 6.

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 run a bittorrent tracker, which maintains a list of seeders for the torrents that are created and shared in MusicDAO. To keep the network decentralized, seeders may also be found using a distributed hash ta-

[4] `https://www.libtorrent.org/manual-ref.html#file-format`
[5] `https://github.com/knowm/XChange/releases/tag/xchange-5.0.1`
[6] `https://www.binance.com/en/trade/BTC_USDT`
[7] `https://github.com/frostwire/frostwire-jlibtorrent`

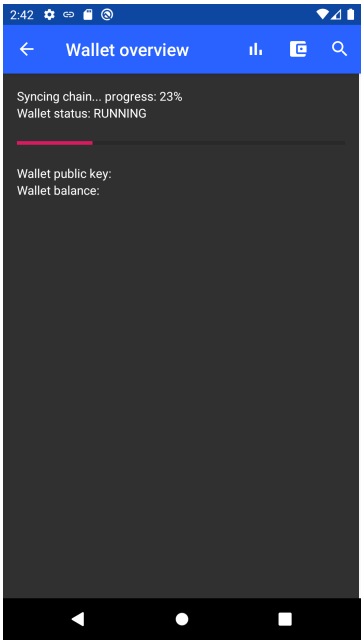

Figure 5.4: Synchronizing with the Bitcoin RegTest environment blockchain

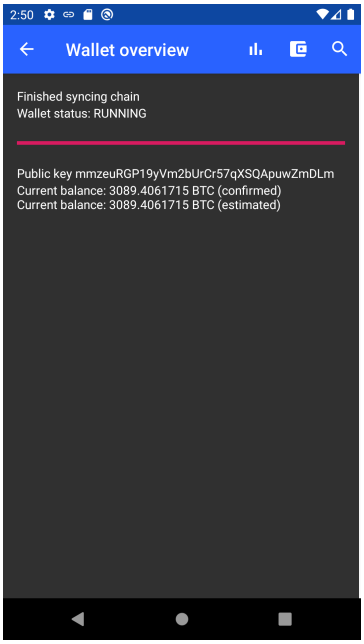

Figure 5.5: Wallet overview and balance after synchronizing

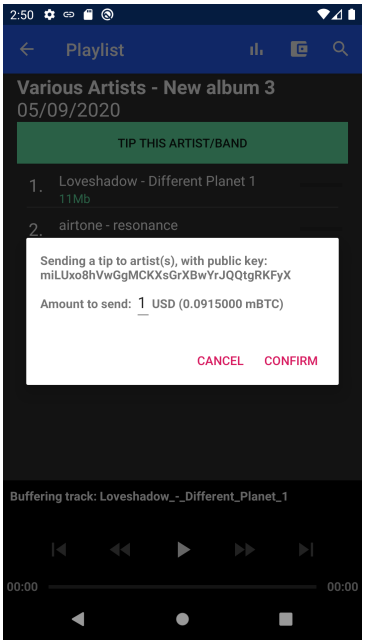

Figure 5.6: Sending a tip to an artist or band

ble (*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.

5.6.2. 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.

5.7. Identity and authenticity

Every device participating in MusicDAO has a unique identity. MusicDAO implements the public-key infrastructure as described in 4.6 using the identity system proposed by Skala (2020). This uses the widely used industry standard *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.

5.8. Peer-to-peer 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 search message (see 4.4) to a few random peers. This asks neighbors 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 query initiator. To disallow packages from endlessly being forwarded on

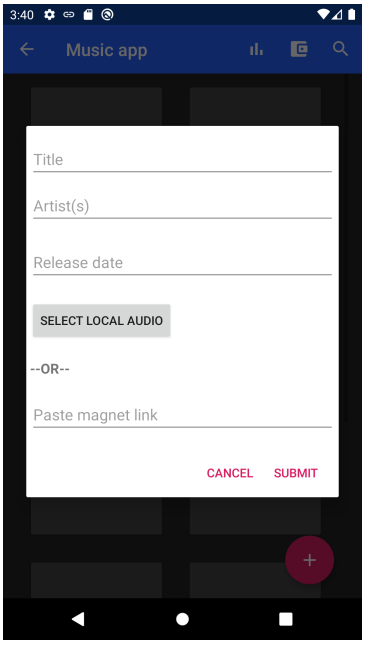

Figure 5.7: Dialog for creating and publishing a new Release

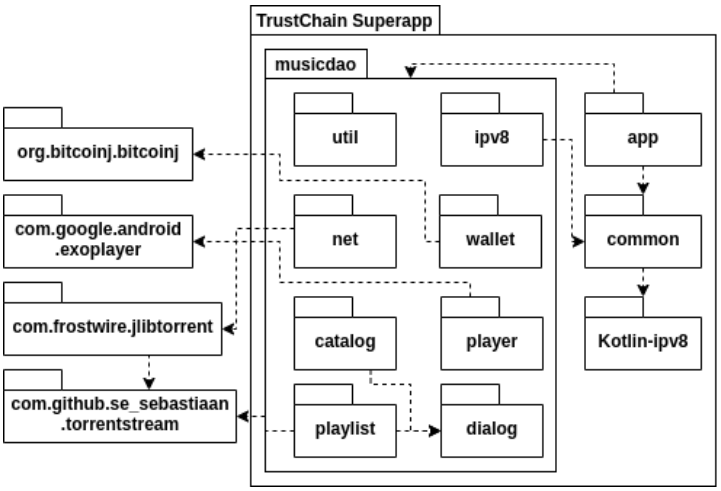

Figure 5.8: Package diagram showing the interaction with external libraries and TrustChain Superapp packages

the network we use a time-to-live property. For details, refer to algorithm 1.

5.9. Peer-to-peer payments to artists

Our system contains peer-to-peer payments where 100% of money goes to artists. We created a public Bitcoin RegTest environment[8] 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.

5.10. Gossip protocol for content popularity

5.11. 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[9] and the mocking library Mockk[10]. The code coverage of these unit tests can be seen in 5.2.

Code coverage is measured by Android Studio 4[11]. 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

[8] https://developer.bitcoin.org/examples/testing.html#regtest-mode
[9] https://junit.org/junit4/
[10] https://mockk.io/
[11] 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 5.2: Code coverage overview

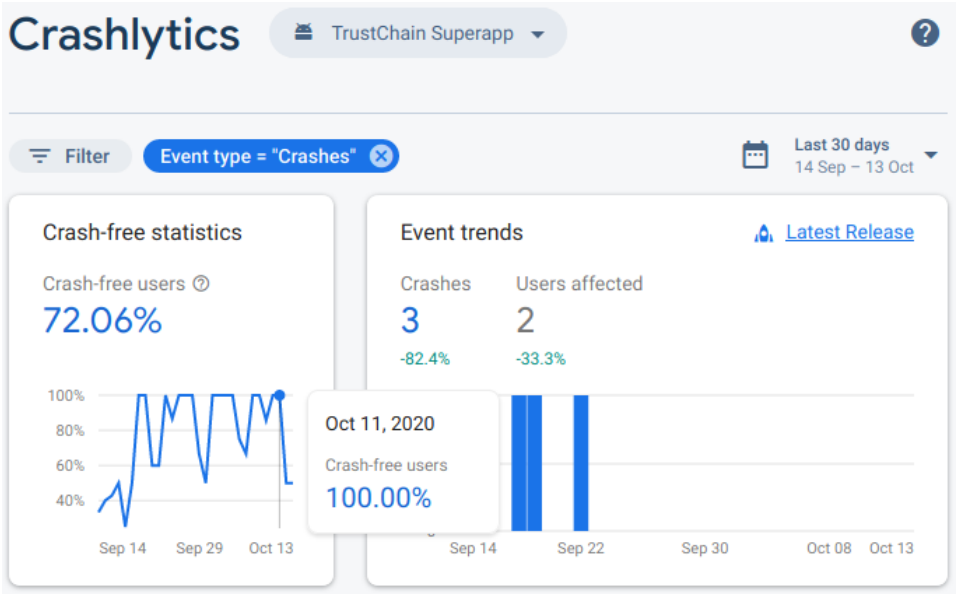

Figure 5.9: Firebase Crashlytics crash reporter; this figure shows the crashes over time

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.

All tests are run in an automated testing environment using Github Actions[12]. 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.

[12] `https://github.com/features/actions`

6

Evaluation

We evaluated the viability of our DAO as a Big Tech alternative. An ideal evaluation would be by measuring the impact of a large-scale (millions of users) and long term deployment of MusicDAO in terms of artist income and artist happiness. However, the time and scale required is not feasible in the scope of this thesis. Instead, we deploy MusicDAO on a smaller scale (20 Android devices, 11 users). We measure performance of money transaction flow, music streaming and search latency in supervised and unsupervised experiments. These measurements determine the responsiveness of our infrastructure in small network sizes. By comparing latency and throughput in several network sizes, we can make conclusions of the performance of our infrastructure in larger networks.

In an *unsupervised* experiment, all test subjects are given the MusicDAO application and are asked to stream music and send money to artists, but with no specific rules or guidelines. We have no control over the actions of these test subjects.

In multiple *supervised* experiments, we use 10 Android devices that are under our control. we measure latency and throughput of music data and money transfers, and analyze the impact of network size on these measurements.

6.1. Unsupervised experiment: public release

MusicDAO was released publicly on the Google Play store and 11 test subjects were given the task to download music and send donations to artists. Upon installation and first running MusicDAO, it sends a request in the background to our Bitcoin node, asking for 10 coins in starter money. The responsiveness of our Bitcoin node for transferring starter money is analyzed in 6.1.1.

6.1.1. Automated starter money transactions

We analyze the responsiveness of our bitcoin node by obtaining two datasets, and inspecting their correlation in terms of timing of events.

The two datasets are:

1. App installations per day, as measured by the Google Play console.
2. Transactions from our Bitcoin node with a value of 10 coins, scraped from our blockchain.

When there are x app installations on some day, there should be yx transactions outgoing from our Bitcoin faucet. In the graph shown by 6.1, this is not always the case. On 19-01-2021 there was (at least) one transaction missing. This was due to a server outage where the Bitcoin faucet was running. There are also cases in which there are transactions but no installations. This may be due to multiple app installations from one device on the same day (which is measured as one by the Google Play store console) or app installations outside of the Play store.

6.1.2. Donation money flow

The datasets used to create the plots shown in figs. 6.2 and 6.3 are received by scraping blockchain block data from our Bitcoin node. Fig. 6.2 shows the relation between created blocks and money transacted per

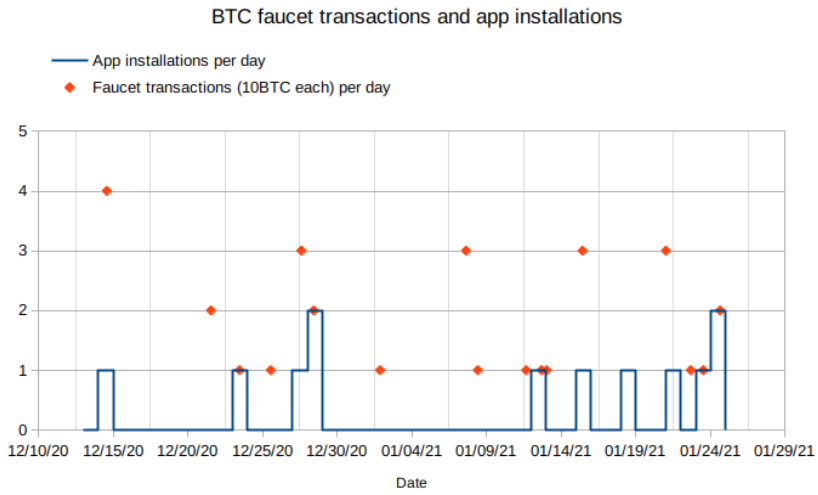

Figure 6.1: Graph showing the relationship between faucet transactions of 10BTC and app installations. Every app installation should correspond to at least one faucet transaction.

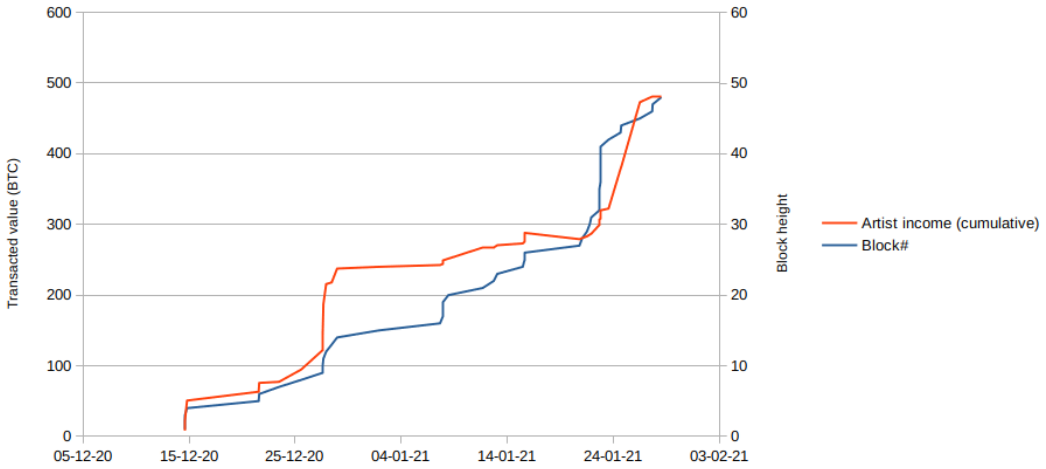

Figure 6.2: Artist income (cumulative) over time and block creation

block (cumulative). We observe that, during the timeline of the release experiment, nearly 500 BTC has been received by artists.

Note that the Bitcoin blockchain only contains timestamps for blocks, and no timestamps for individual transactions. This means we could not analyze transaction confirmation time in our unsupervised experiment, as this requires the timestamp of creating a transaction. However, prediction and analysis of confirmation times in Bitcoin have already been performed multiple times in recent literature (Kawase & Kasahara, 2017) (Koops, 2018). We observe that artist income is

6.2. Supervised experiments

During several supervised experiments, we were in control of a network of 10 Android devices, of which 5 virtual emulators and 5 real world devices. By performing several experiments, throughput and latency of several actions in this network is analyzed. Performing measurements in different network sizes (2 up to 10 devices), enables making predictions of the scalability of MusicDAO.

6.2.1. Downloading and streaming

6.4 shows the download time of each stage in the bittorrent downloading process. By running 10 runs per network configuration, we inspected the effects of a bittorrent tracker on throughput and handshake time.

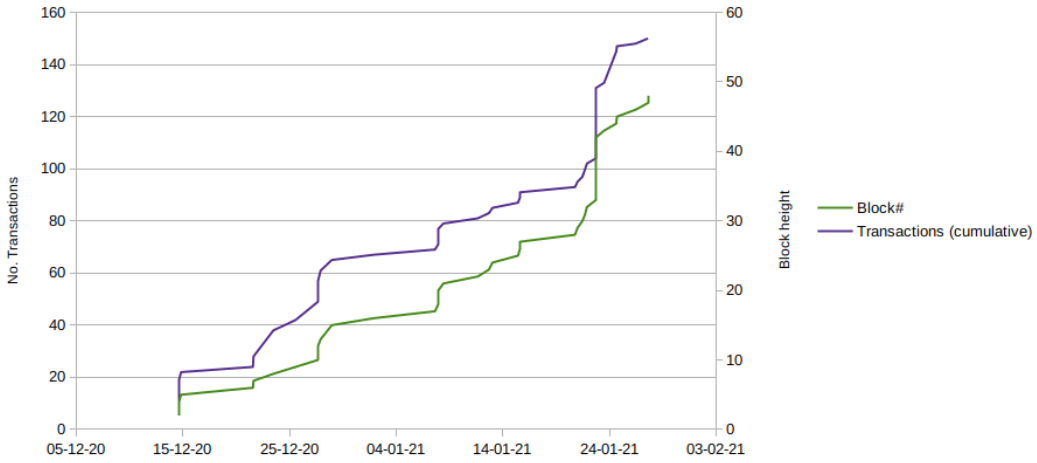

Figure 6.3: Transactions (cumulative) over time and block creation

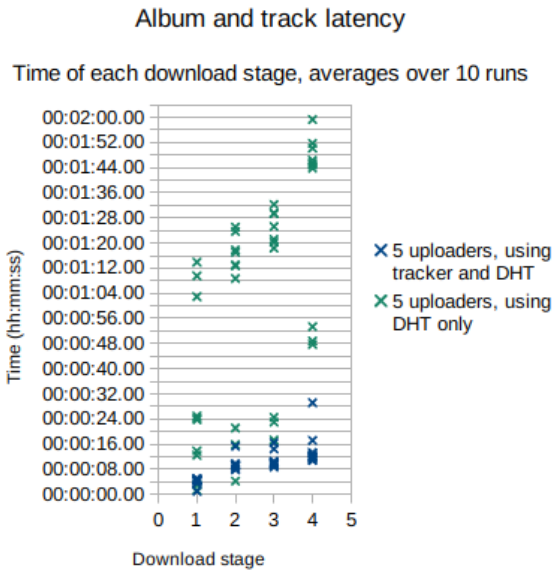

Figure 6.4: Average time spent per download stage. Measured by 20 runs in total, in two different network configurations.

The 4 different download stages marked in this figure are as follows.

1. Time to receive metadata (including establishing handshake)
2. Time to fill stream buffer
3. Time to download first track
4. Time to download full album

The measurements show that a bittorrent tracker significantly reduces transfer times, for a small bittorrent swarm with 5 seeders. The largest difference is in the *fetching metadata* stage, during which the device under test must find and connect with seeders. Discovering seeders over DHT requires asking multiple peers, and as such requires more time and messages before the download can start. Once the download starts, the runs using a tracker also reach significantly higher throughput, as the tracker assists the device in finding more seeders and healthier seeders. We found that the major factor slowing down download stages when using DHT only is the NAT puncturing stage, where devices try to connect to each other when there are one or more NAT devices in between. As expected, finding peers over DHT is also slower than via a tracker, however

The datasets used to create the plots shown in figs. 6.2 and 6.3 are received by scraping blockchain block data from our Bitcoin node. Fig. 6.2 shows the relation between created blocks and money transacted per block (cumulative). We observe that, during the timeline of the release experiment, nearly 500 BTC has been received by artists.

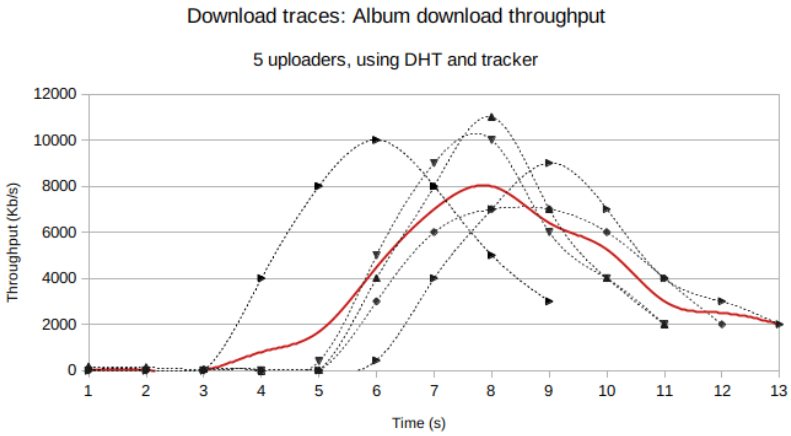

Figure 6.5: 5 traces of download throughput, downloading an album of around 38 Mbs. Measured with Nokia 7.

the time taken to create a handshake during NAT puncturing has a much larger effect on the peer-to-peer connecting time (the time taken to create a reliable connection for data transfer).

6.5 shows 5 traces of downloading a 38 Megabyte album. The red line shows the moving average over these 5 runs. The slow-start nature of bittorrent can be observed here. Roughly the first 5 seconds are used for fetching the bittorrent metadata (see also 6.4).

Note that all devices evaluated in experiments 6.4 and 6.5 use Bittorrent Local Peer Discovery. This means that some of the data transfers may be over local area network, which reaches considerably higher throughput than regular transfers.

6.2.2. Content discovery

Fig. 6.6 shows measurements of an Android device discovering content, after running MusicDAO for the first time. More specifically, it is a measurement of music metadata, received as TrustChain blocks. All participating devices are configured as follows: Every device sends a random block to a random peer every 5 seconds. A Nokia 7 Android device ran the MusicDAO in an idle state for 5 minutes. The app measured the amount of content metadata discovered every 2 seconds.

Mathematical model for evaluating experiment data

For evaluating content discovery over time, we compare three different experiments with a mathematical model for expected discovered items over time.

- Gossip interval $i = 5$ seconds
- Total items to discover: $R = 50$

We device *hit chance* as such: every interval i, the hit chance C to receive any item is

$$C = 1 - (\frac{p-1}{p})^p$$

with p as the number of peers. This is according to *Montmort's Matching Problem* (de Montmort, 1713). We compute C for the network sizes $n = 2, n = 4, n = 10$ with

$$p = n - 1$$

The expected value for the cumulative amount of unique items discovered after x iterations is

$$E[R] = R(1 - (\frac{R-1}{R})^{Cx})$$

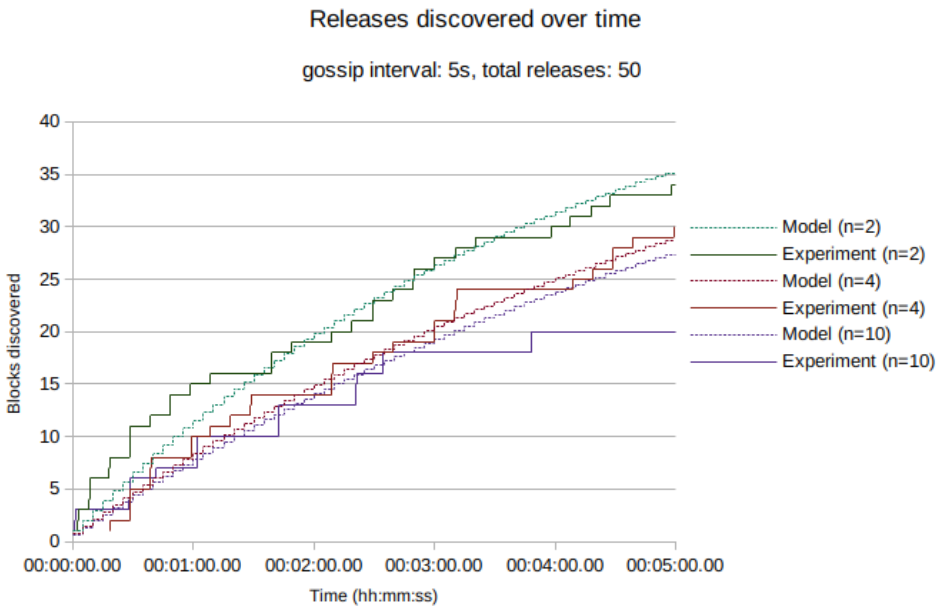

Figure 6.6: Measurements of content discovery: releases discovered after a fresh installation. n: network size (amount of Android devices)

where

$$x = \frac{1}{i}$$

This expected value $E[R]$ over time is plotted in 6.6 as Model, for three different network sizes n.

In 6.6 we can see a correlation between model and experiment for network sizes $n = 2$ and $n = 4$. For $n = 10$ the experiment and model correlate until 3 minutes into the experiment. The reasons for this are for now not clear. More investigation and more experiments on larger networks are necessary to make conclusions.

6.2.3. Random access latency

Random access latency is evaluated through measuring search latency. Search latency is the round-trip time between initiating keyword search and receiving metadata for the release. During this experiment, the content that was searched for was present in all devices except the one under test. The aforementioned distributed search algorithm (alg. 1) is used with the following parameters.

- $maxPeers = 20$
- $ttl = 1$

This means that a maximum of 20 neighbours are contacted when performing a search, and that search is not recursive; meaning only neighbours are contacted (not neighbours of neighbours).

For two different situations, 10 runs are done and the latency and average latency are plotted in 6.7. We observe that for these 20 runs, latency is $< 1s$. Another observation is: increase in network size (10 versus 2) does not negatively affect latency. Current music streaming systems have a search latency of roughly $< 2s$ in ideal situations.

6.3. Devices behind NATs

During experimentation, some of the Bittorrent traffic on Android devices were blocked by Network Address Translators (NATs). MP3 transfers over Bittorrent were slowed down by this. The NAT Port Mapping Protocol (NAT-PMP) was used to be able to establish connections between different devices. NAT-PMP establishes connections using port scanning and port forwarding. However, this is a lengthy process: we observed that establishing such a connection usually takes more than 2 minutes. Analyzing and improving this process should be investigated in future distributed systems research.

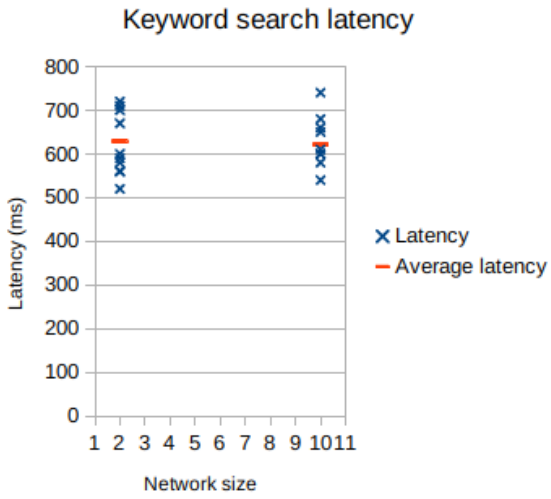

Figure 6.7: Search latency of performing keyword search

6.4. Transaction fees

6.5. Missing features

- Artist Income Division Algorithm

6.6. Scalability

Scalability beyond 20 nodes

6.7. Music publishing protocol

Music publishing currently requires a block signing/agreement from one other peer, before it is added to the TrustChain and sent to other peers.

6.8. Bitcoin node

The experiments made use of a bitcoin node, which ran on a dedicated server 24/7.

6.9. Bootstrap node

7

Conclusion

7.1. Main characteristics of our presented framework

- Leaderless, fully distributed system
- Fairness, middleman-free, 100% money to artists
- Transparency in database layer (trustchain) and source (open source)
- Identity system with public history, chains-of-trust
- Peer-to-peer streaming
-

7.2. Generality of our AI DAO framework

Applications in other domains: decentral markets, youtube alternative,

7.3. Future research directions

- Actual AI used for decision making
- Content moderation by voting and/or AI
- Continuous, democratic code evolution (based on voting protocol), bountysource.com, secure plugin system related work
- Artist identity/passport, relate to Self-Sovereign Identity related work (SSI)
- Towards fully fair, transparent and open democratic systems for the common good (discuss what the holy grail is of a Robot Economy)

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.

BBC. (2019). Spotify sued over 'billions of eminem streams'. Retrieved October 25 2020, from https://www.bbc.com/news/technology-49436077

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. (2014). Daos, dacs, das and more: An incomplete terminology guide. *Ethereum Blog*. Retrieved November 4 2020, from https://blog.ethereum.org/2014/05/06/daos-dacs-das-and-more-an-incomplete-terminology-guide/

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

Cohen, B. (2002). Bittorrent protocol specification v1.0. *WWW, June*.

de Montmort, P. R. (1713). *Essay d'analyse sur les jeux de hazard...* J. Quillau.

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

DigitalMusicNews. (2018). What streaming music services pay (updated for 2020). Retrieved 29 October 2020, from https://www.digitalmusicnews.com/2018/12/25/streaming-music-services-pay-2019/

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

et al., D. F. (2014). Bypassing censorship: A proven tool against the recent internet censorship in turkey. In *2014 ieee international symposium on software reliability engineering workshops* (pp. 389–394).

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

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 ....

Kawase, Y., & Kasahara, S. (2017). Transaction-confirmation time for bitcoin: A queueing analytical approach to blockchain mechanism. In *International conference on queueing theory and network applications* (pp. 75–88).

Koops, D. (2018). Predicting the confirmation time of bitcoin transactions. *arXiv preprint arXiv:1809.10596*.

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

Marozzo, F., Talia, D., & Trunfio, P. (2020). A sleep-and-wake technique for reducing energy consumption in bittorrent networks. *Concurrency and Computation: Practice and Experience*, *32*(14), e5723. Retrieved from https://onlinelibrary.wiley.com/doi/abs/10.1002/cpe.5723 doi: 10.1002/cpe.5723

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

Meier, L. M., & Manzerolle, V. R. (2019). Rising tides? data capture, platform accumulation, and new monopolies in the digital music economy. *New Media & Society*, *21*(3), 543–561.

MidiaResearch. (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. (2012). *Media without censorship (censorfree) scenarios.* Retrieved 09 November 2020, from https://tools.ietf.org/html/draft-pouwelse-censorfree-scenarios-02

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.

Rayna, T., & Striukova, L. (2009). Monometapoly or the economics of the music industry. *Prometheus*, *27*(3), 211–222.

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).

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

Stiglitz, J. (2019). Market concentration is threatening the us economy. *Project Syndicate*.

Townsend, K., Cufí, C., Davidson, R., et al. (2014). *Getting started with bluetooth low energy: tools and techniques for low-power networking.* " O'Reilly Media, Inc.".

Trichordist, T. (2014). The streaming price bible – spotify, youtube and what 1 million plays means to you! Retrieved 22 October 2020, from https://thetrichordist.com/2014/11/12/the-streaming-price-bible-spotify-youtube-and-what-1-million-plays-means-to-you/