Fairness and Freedom for Artists: Towards a Robot Economy for the Music Industry

Tim Wissel

Fairness and Freedom for Artists: Towards a Robot Economy for the Music Industry

by

Tim Wissel

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

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

Preface

Tim Wissel
Delft, October 2020

Contents

1 Introduction

The centralization of power on Internet platforms cause unfairness in many industries. The technological advancements of the Internet has mainly lead to income polarization. The explosive increase in revenue on Internet platforms has mainly lead to higher profit margins for Big Tech instead of the core contributors on these platforms. During the last decades, we observe the trend of *platformization*: a shift of economic activity from happening on a wide range of companies to a few major platforms run by Big Tech corporations. This trend is highly susceptible to the rise of monopolies and oligarchs. This results in small and untraceable payouts to the core contributors on such platforms. The users of centralized platforms are affected by decisions that are made based on business strategies instead of democratic procedures.

Platformization already has a strong effect on the music industry, in which the succesful music streaming services have an immense amount of power. 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 and labels take large revenue cuts of up to 40%. The processing of royalties through many intermediaries is unclear on purpose. Artists can also suffer from political interference in these companies, reducing their freedom of speech. This thesis aims to distribute the power from centralized streaming platforms to listeners and artists, with the goals that artists (1) obtain a fair income and (2) are freed from censorship.

The main contributions of this thesis are:

1. A novel framework as an alternative for Big Tech: the robot economy in software;
2. A partial implementation of this framework: a fully decentralized Android music streaming application *MusicDAO* which attempts to liberate artists and consumers from powerful intermediaries.

MusicDAO implements a few key components of our framework: P2P leaderless infrastructure, resilient communication, trustless content sharing/exploration and a trustless monetary system (see 1.1). We design, implement and evaluate the MusicDAO system. We perform supervised experiments with 10 Android devices to test the responsiveness and latency of its main features. Aditionally, we release the application publicly to 50+ users and measure the monetary flows from users to artists over time. It is available on Google Play[1] and its code is open source[2].

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. The consumer choice is diminishing due to the power of oligarchs and monopolies. A few Big Tech corporations are gaining increasing power in the the surface on which market exchange takes place. These corporations take a large cut in revenue streams which majorly affects income for creators of content and services. Examples are Uber, Ebay and Spotify. Revenue streams towards the core contributors on a platforms are opaque on purpose. The software and algorithms powering these platforms are also a

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

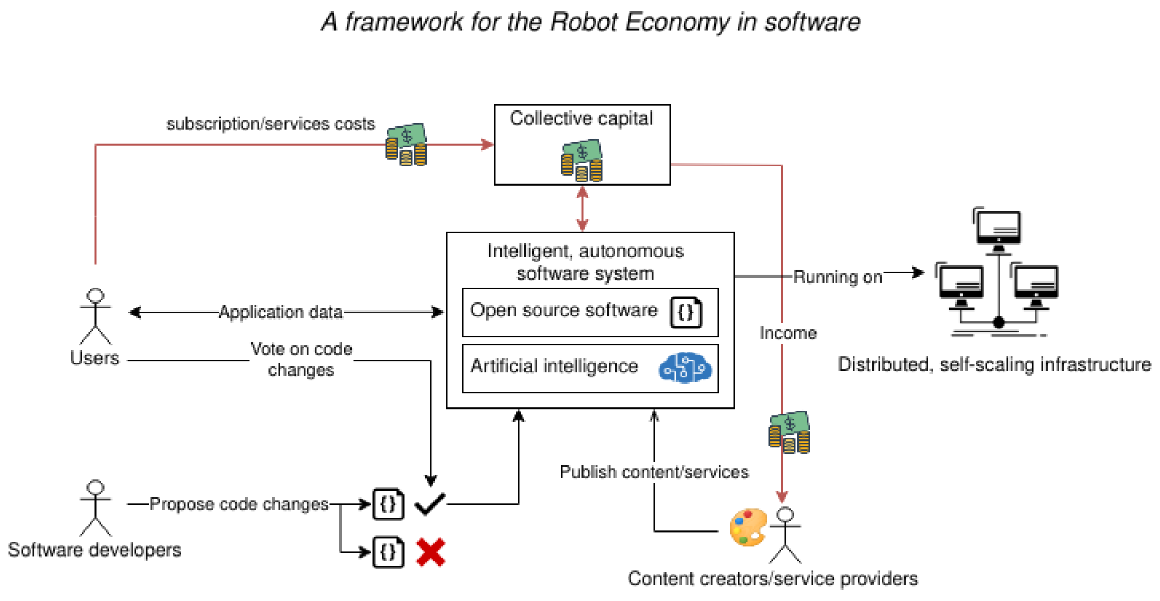

A framework for the Robot Economy in software
subscription/services costs
Collective capital
Intelligent, autonomous software system
Open source software
Artificial intelligence
Running on
Income
Distributed, self-scaling infrastructure
Users
Application data
Vote on code changes
Software developers
Propose code changes
Publish content/services
Content creators/service providers

Figure 1.1: A robot economy in software: democratic and autonomous software

black box to creators and consumers. Furthermore, users have negligible influence on the future of these platforms. As explained by (Stiglitz, 2019), the funcamental problem is 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. Towards a robot economy

An alternative for Big Tech is building a robot economy in software. Our novel vision of a robot economy follows recent theoretical groundwork by Arduengo and Sentis (2020): In a robot economy, intelligent robots play a key role, by performing economic operations autonomously. Robot tasks are driven by artificial intelligence, and cooperate with humans. While robots already take an active part in society today, the key difference in this vision is that robots have internal funds (which may be money, tokens or other assets) and can perform transactions on their own. Our work sets the first steps towards a robot economy in software by building and evaluating a few key components of this theory. Software as a robot economy is a service that runs autonomously, with which humans interact. Humans can spend funds, perform decisions or interact with data, while the software is run by robots.

A robot economy in software can have large influence: it can replace a company, or even a full value chain, Our framework for a robot economy in software allows designing and implementing *infrastructure for the common good*: software that is governed by its key users and contributors, and as such leads to more fairness on Internet platforms. In a traditional software system, a company decides the parameters and functions of software. The software is run on company infrastructure only, which creates central points of failure. In a software system in the robot economy, its parameters and functions are voted on by its user base in a democratic voting process. It is run on distributed infrastructure, so is not susceptible to central points of failure. Since to the introduction of blockchains and crypto-currency, this voting process can be automated and transparent. Robots make intelligent decisions based on data and value given by humans. Such an implementation can replace entire value chains of industries.

As seen in 1.1, the central component of the Robot Economy in software is an intelligent autonomous software system using a DAO. Every participant can vote on changes of the inner workings of the software using a democratic voting protocol.

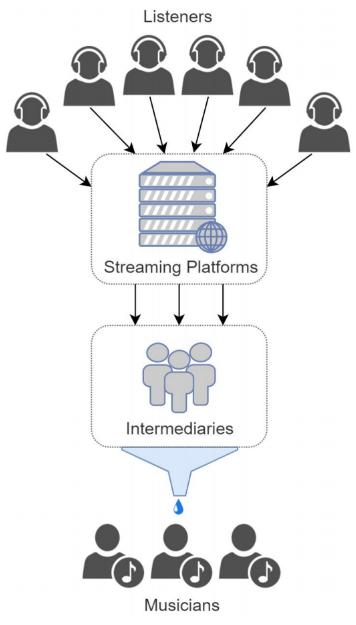


Figure 1.2: Artist compensation inconsistency

A full robot economy aims to have the following key characteristics:

- Automated;
- Transparent;
- Fair;
- Democratic;
- Open (permissionless);
- Leaderless;
- Self-evolving.

To accomplish these key characteristics in a software system, we envision the main building blocks to be as described in fig. 1.1.

1.3. 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.

Component	**Focus in MusicDAO**
Peer-to-peer leaderless infrastructure	✓
Resilient communication	✓
Trustless content sharing and exploration	✓
Trustless monetary system	✓
Democratic user engagement	×
AI for decision making (robot tasks)	×
Continuous code evolution and distribution	×

Table 1.1: The main components to achieve a robot economy in software

1.4. Proposed solution: MusicDAO

This thesis proposes an alternative technology from Big Tech streaming services. We observe that most functions of music streaming systems are already completely automated. 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 automation itself cannot do” (see 3.2). 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. 100% of music revenue goes to artists.

MusicDAO contains the following key 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.

In a real-world experiment with Android phones, we tested the feasibility of such a phone-to-phone infrastructure-less system. In a public release experiment, MusicDAO is published to the crowd via the Google Play Store[3], and is installed on more than 50 devices. We evaluated all the money flow from its users towards the artists using a public blockchain. We evaluated the responsiveness of our decentralized infrastructure using supervised experiments, in which we measured the effect of network size on the latency of the key functionalities of our system.

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

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 82%, and the top 3 labels 70% combined market share. These major corporations have huge power over both the producer and consumer 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 on streaming platforms are a black box to the user. This gives streaming and playlist 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. According to MidiaResearch (2020), the top 5 streaming services have 82%, and the top 3 labels 70% combined market share (see 2.2, 2.1). The core problem follows: “A large number of sellers (musicians, singers, bands) are in interaction with a very small number 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 advantageous network infrastructures 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): the routing of all music and money flow over centralized platforms.

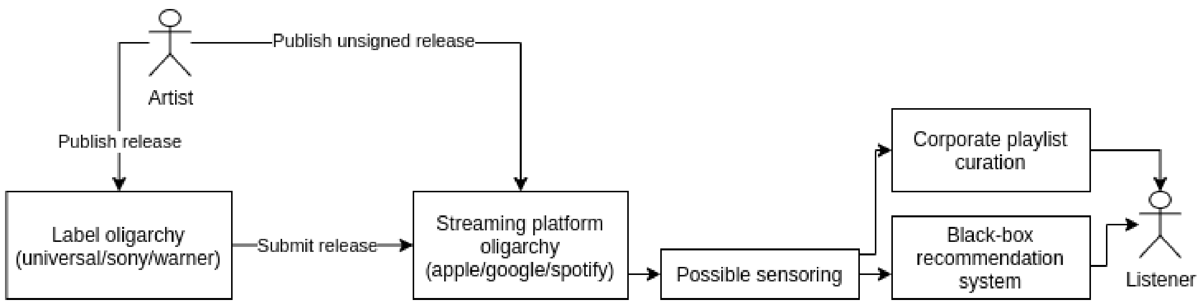

Artist
Publish unsigned release
Publish release
Label oligarchy (universal/sony/warner)
Submit release
Streaming platform oligarchy (apple/google/spotify)
Possible sensoring
Corporate playlist curation
Black-box recommendation system
Listener

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

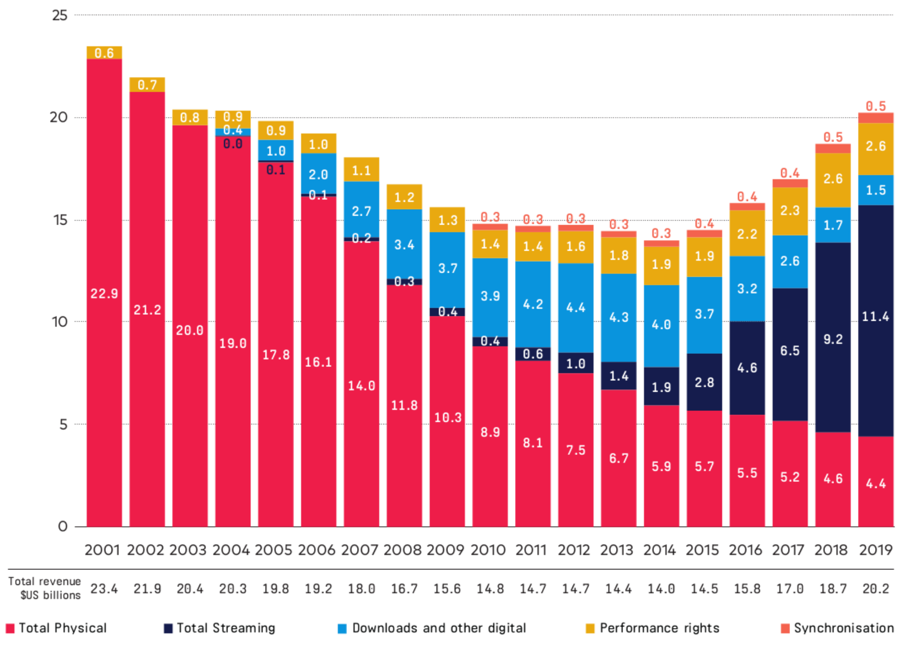


Figure 2.2: Global recorded music industry revenues (in Billion USD) (source: IFPI (2020))

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 (source: MidiaResearch (2020))

Around the 2010s, the attention started to turn again to a new industry: on-demand streaming of music. This started a major transition in the last 10 years. The financial sources of the music industry changed rapidly as seen in fig. 2.2. According to Friedlander (2020), streaming accounted for 79% of music revenue (excluding concerts and merchandise) in the US in 2019, compared to only 15% in 2012. 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.

Major music label	**Market share**
Universal Music	31%
Sony Music	21%
Warner Music	18%
Total	**70%**

Table 2.2: Global market share of the top 3 music labels, measured by volume of recorded music (MidiaResearch, 2020)

2.2. Monopoly power of centralized platforms

The infrastructure of current internet applications are increasingly moving towards *platformization*. In the context of music, subscriptions and purchases are majorly happening on a few central platforms, such as Spotify or Apple Music. In essence, platforms are taking control of “the surface on which the market exchange take place” (Andersson Schwarz, 2016) with digital distribution and network effects enabling an increasing centralization of power. This phenomenon is related to IT gatekeeping: tying access of content to a specific internet service. An example of this is the release of the album *The Life of Pablo* in 2016, which was contracted to only be played on one platform, Tidal.

In 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[1], who published songs about democracy and social issues in China. All of China’s main streaming sites removed his songs. In 2019, Apple Music also removed content from their platform by singer Jacky Cheung, who referenced the tragedies of Tiananmen Square in his songs[2]

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).

Aside from a monopoly, Rayna and Striukova (2009) interpret these platforms as *monopsomies*. 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” [3].

2.3. 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 2.3). 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) notice that “major record labels have substantial ownership stakes in Spotify”. Meier and Manzerolle (2019) also point out that Big Tech platforms have shown to make lucrative deals with major labels, paying them millions of dollars up front.

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. Only a small portion of all music revenue lands in the hands of artists, as seen in fig. 2.3. This graph shows a large portion of revenue declared as *platform costs* for streaming services, however it is unclear how much of these costs are for actual infrastructure and services and how much is for profit. According to investigations by aCooke (2018) and ReCode (2015), the revenue cut of Apple Music and Spotify is between 25% and 40%. The largest revenue cuts are for independent musicians, which makes it very difficult to make a stable income while not contracting with the label oligarchy and music streaming oligarchy (see 2.1). 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

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

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

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

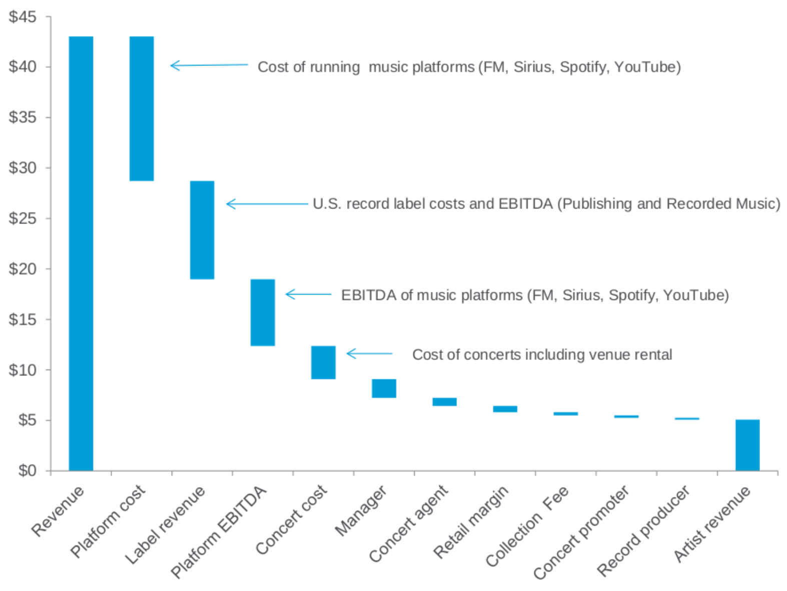


Figure 2.3: Artist revenue compared to other intermediaries (source: Bazinet et al. (2018))

	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 2.3: 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).

Bloomberg[4] shows that 152,094 Spotify subscriber streams generate only $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).

2.4. Financial transparency

The actual overhead of the music industry is unclear, as the flow of money towards artists is either not public or lacks detail and explanation. Contract details and royalty payments are vague on purpose. This is part of the business model of music stakeholders. According to Music (2015), “Despite streaming services paying the same percentage of their revenue (70 percent) to rights holders as an iTunes download sale, low payouts and many intermediaries are creating concerns” (Music, 2015).

The complex flow of intermediaries results 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 infrastruc-

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

ture needed to make sure songs are licensed and musicians are paid". Royalty payments are generally also unnecessarily time-consuming: it can take 6 to 12 months to arrive at artists (Music, 2015).

It appears that a large amount of royalty revenue flows outside of the artists, in a "black box" (Music, 2015). The reason is that there is no industry-wide system for declaring rightful owners of music. This problem is even more significant as there is no incentive for the major labels and major streaming services to improve this situation. These companies deliberately obfuscate royalty processing, or use outdated and unnecessarily complex systems.

2.5. Profit-driven recommendation engines

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

Musicians and record labels are increasingly 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 major 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. Pelly (2017) describes that "[the Spotify front page] is tightly controlled by Spotify's staff and dictated by the interests of major labels, brands, and other cash-rich businesses who have gamed the system".

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 playlist curation is performed 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 (financial) pressure from artists and labels, demanding their work to be visible on her playlists. This makes the music industry even more unfair for smaller artists with less power.

2.6. 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 in a fully distributed network, without a company or a single point of responsibility, 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 on 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 and DAO

The combination of intelligent agents and DAO technologies allows building an autonomous, decentralized and self-evolving software system. This may present the first ingredients towards a robot economy in software. In a system on the boundary of DAO and AI technologies, humans are still involved with performing some tasks and governing, but intelligent agents can perform other decision making tasks on their own. Decision making by robots can be done using a wide range of state-of-the-art algorithms and techniques shown in fig. 3.1.

One implementation of intelligent agents acting autonomously is CloudOMate (Jaspers Focks, van Veltom, Wigmore, & Nguyen, 2018). This is a software system that runs autonomously and is self-replicating: it can extend its network infrastructure on its own by contacting hosting companies. This can be connected to an AI algorithm which learns when an expansion of network resources is necessary. It has internal capital that is used to pay service fees to hosting providers.

In our context, we envision human tasks to be e.g. (1) creating music and giving feedback on music (through e.g. donations), (2) making decisions on the subscription protocol and the payouts to artists, or

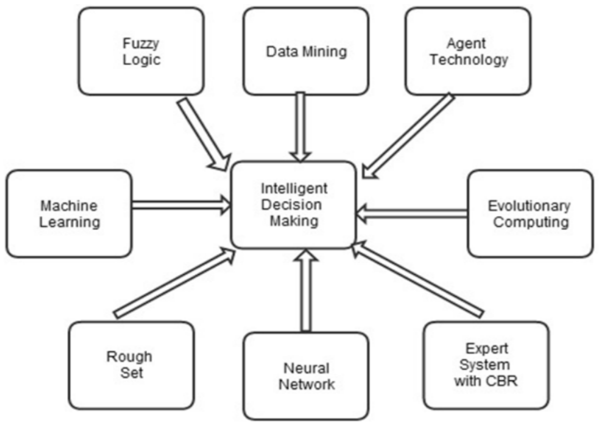


Figure 3.1: Intelligent techniques in decision making (source: Das (2016))

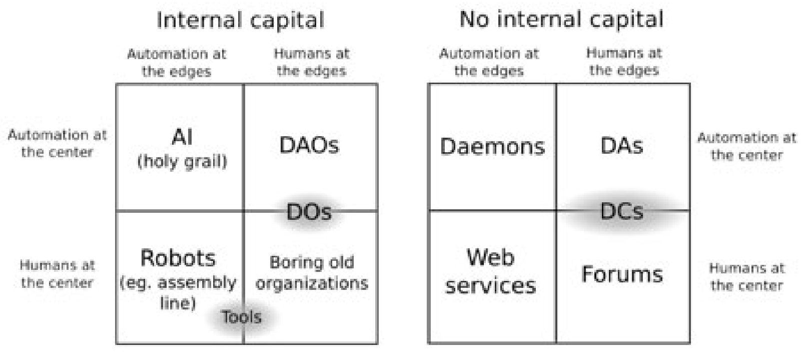


Figure 3.2: Decentralized Autonomous Organization, in comparison to other organizational structures. In a DAO, activity is performed by humans at the edges, automation at the center. It is an interplay in which robots and humans perform tasks. Source: Buterin (2014)

(3) governing evolution of the software. ’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.

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).

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. Its monetary infrastructure is displayed in fig. 3.3. 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.

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, and to cooperate in its permissionless DAO (see fig. 3.4).

Musicoin is a blockchain platform without intermediaries that focuses on income for independent artists.

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

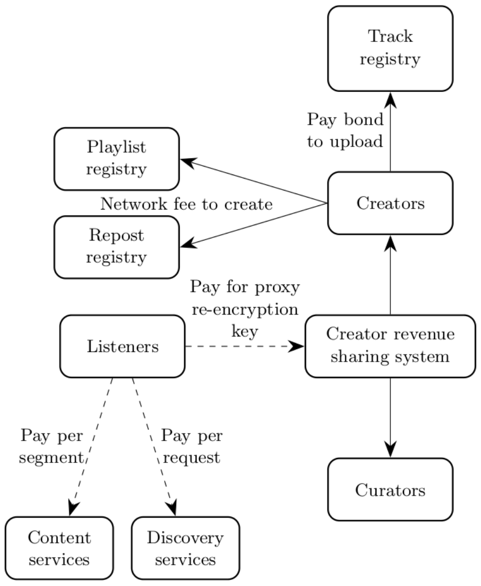


Figure 3.3: Monetary flow in Audius (source: Rumburg et al. (2018))

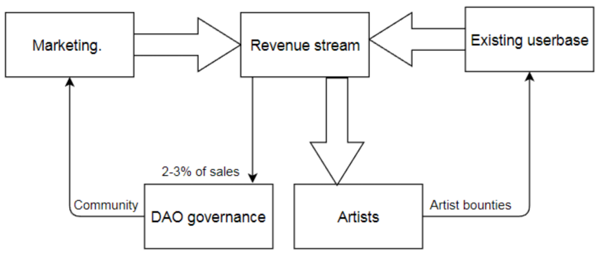


Figure 3.4: Effect of DAO governance system on revenue streams in Opus (source: (Jia et al., 2016))

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 higlhy volatile.

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 stable currency.

3.4. Financial transparency in the music industry

As described in sec. 2.4, there is a serious issue in the music industry affecting artists globally: the (by design) inaccurate, complex and slow royalty payments. There are existing decentralized frameworks and protocols to solve this problem.

Payments of royalties to artists can be described in transparent, public and immutable records on a blockchain. In addition, smart contracts can be used to automate payments (Buterin et al., 2014). Smart contracts are self-executing (non-ambiguous) 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, on a distributed virtual machine (VM) such as the Ethereum VM.

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 network without servers. It follows the concept of super apps (Huang,

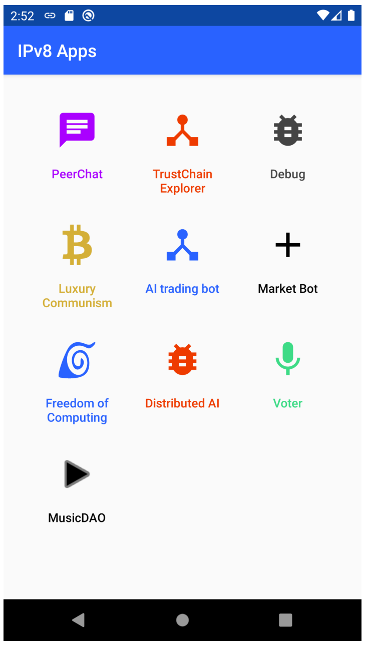
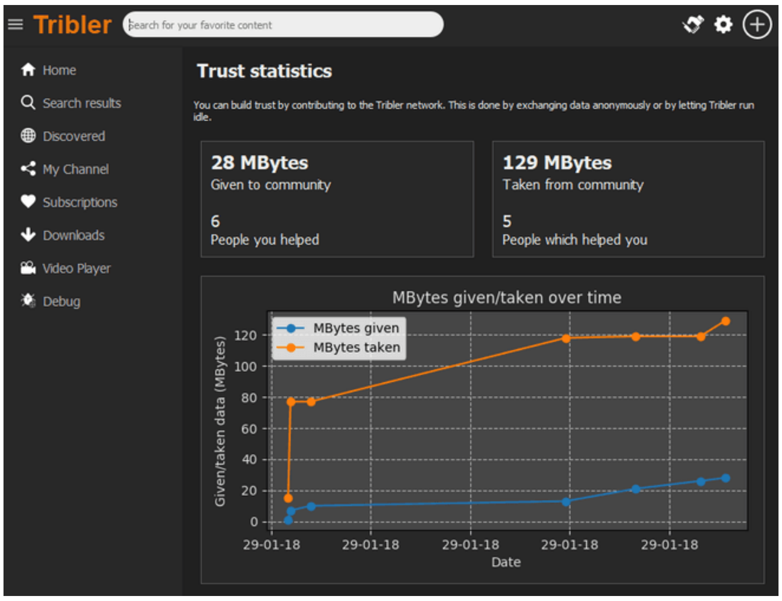


Figure 3.5: Trustchain-Superapp overview



Trust statistics

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

28 MBytes
Given to community

6
People you helped

129 MBytes
Taken from community

5
People which helped you

Figure 3.6: Tribler desktop interface, showing the bandwidth incentive system overview

2019), meaning that it contains many mini-apps which use the same networking interface. The Superapp is an Android app as seen in fig. 3.5. 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. 3.7). 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 and votes is done using the TrustChain DLT (Otte et al., 2017). This voting app is an important basis for democratic decision making. It could be extended to include continuous, automated code evolution: by voting on code patches or changes, the community can decide on new features of the application. This protocol is based on proof-of-identity instead of proof-of-stake, as no tokens are involved.

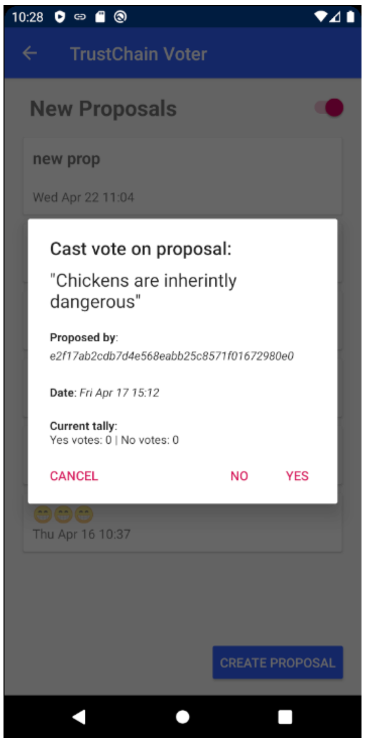
3.6. Decentralized content delivery networks

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.

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. BitTorrent originally relied on trackers to perform peer discovery, and trackers can become a central point of failure. However, since the introduction of the DHT protocol (Loewenstern & Norberg, 2008), finding peers in the network can be done by querying any known peer, which makes the network more decentralized.



Cast vote on proposal:

"Chickens are inherintly dangerous"

Proposed by: *e2f17ab2cdb7d4e568eabb25c8571f01672980e0*

Date: *Fri Apr 17 15:12*

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

CANCEL NO YES

Figure 3.7: *Distributed democracy* voting mini-app, showing proposals

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. It uses tit-for-tat as a method of efficiently downloading files. However, a challenge in BitTorrent 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 (IFPS), 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. When comparing IPFS and BitTorrent, a notable difference between IPFS and BitTorrent is that IPFS makes a distinction between 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. File hosting incentive algorithms

In a DAO, the party responsible for hosting and spreading of files is not well-defined. To tackle the tragedy of the commons, participants should have an incentive to contribute their computational and bandwidth

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

resources for the network to be sustainable and responsive. One example incentive system is bandwidth tokens (de Vos & Pouwelse, 2018) as part of the Tribler system. Another is the FileCoin

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. 3.6.

FileCoin (Benet & Greco, 2018) is an addition on top of the distributed file storage system IPFS. It is a distributed incentive protocol which rewards users for hosting files in the network. In essence, there is a decentralized market over which cryptocurrency (FileCoin) is transacted. Supply and demand of files in the network determine the reward value in terms of FileCoin. The mechanism for obtaining these coins is somewhat similar to the mining protocol in Bitcoin, but uses *proof-of-storage* instead of *proof-of-work* as a way to determine block rewards.

4 Design

Much is still unknown in the research area of the Robot Economy in software. This thesis aims to create more knowledge on this theory by attempting to build a proof-of-concept of such intelligent system.

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. It implements a few key components (see table 4.1) of our framework for the robot economy presented in fig. 1.1. 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.

We build upon the theory of a Decentralized Autonomous Organization (DAO) (see sec. 3.1). This is a piece of software that lives autonomously but also heavily relies on humans to do certain tasks. A DAO has internal capital: some kind of value that is used internally to pay humans or robots to complete tasks.

Robot Economy component	**Focus in MusicDAO**
Peer-to-peer leaderless infrastructure	✓
Resilient communication	✓
Trustless content sharing and exploration	✓
Trustless monetary system	✓
Democratic user engagement	×
AI for decision making (robot tasks)	×
Continuous code evolution and distribution	×

Table 4.1: The main components to achieve a robot economy in software (repeated)

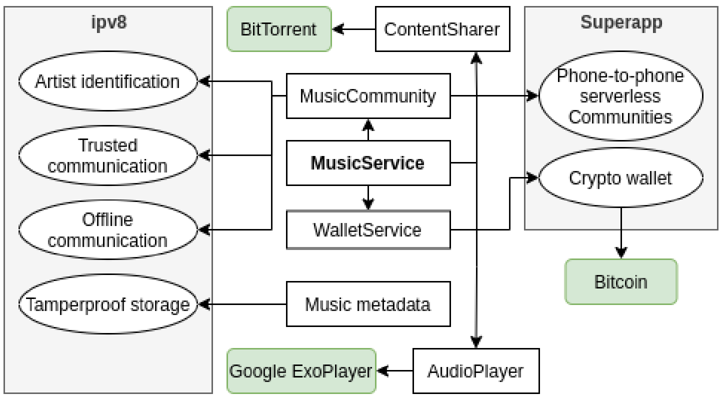


Figure 4.1: Architecture overview, with in green the external libraries. MusicService is the central component in our system

4.1. Zero-cost autonomous music industry

We design a system that takes important 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 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. Zero-server phone-to-phone application

Traditional Internet services are built around a single server or a set of servers (fig. 4.2). We design a network which only consists of mobile phones (fig. 4.3). 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.

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 2.2. 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:

4.3. Phone-to-phone connectivity

MusicDAO is designed to have a network consisting of only mobile phones (fig. 4.3). 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

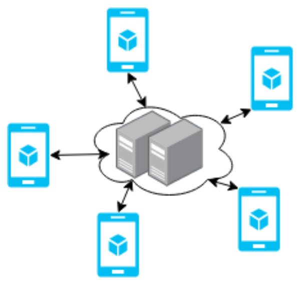


Figure 4.2: Traditional (centralized) Internet service infrastructure

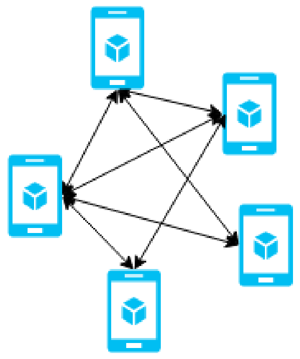


Figure 4.3: Peer-to-peer network of phones, as in MusicDAO

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 a wide range of applications to emerge, all using the same streaming, discovery and payment protocol, but with various different user interfaces and strategies for content filtering. 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. 4.4. 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. 4.5. 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

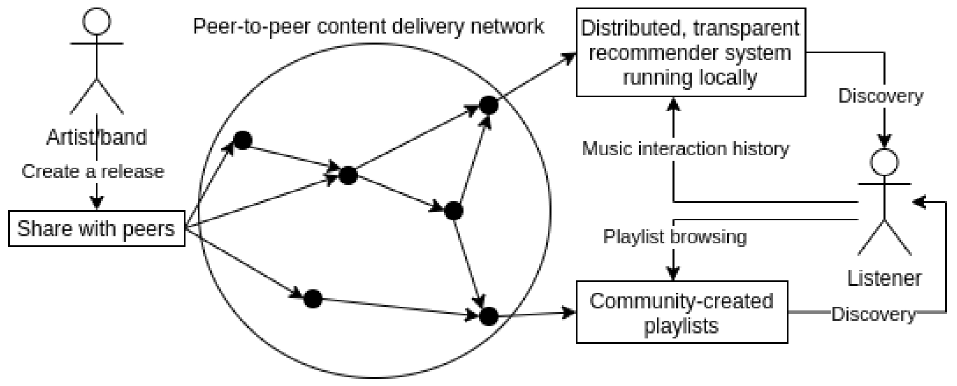

Peer-to-peer content delivery network
Artist/band
Create a release
Share with peers
Distributed, transparent recommender system running locally
Discovery
Music interaction history
Playlist browsing
Listener
Community-created playlists
Discovery

Figure 4.4: Desired music publishing flow using a distributed network

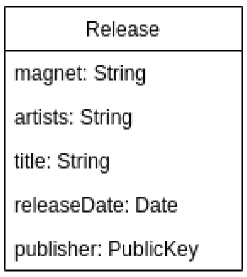


Figure 4.5: Release blocks structure as seen on TrustChain

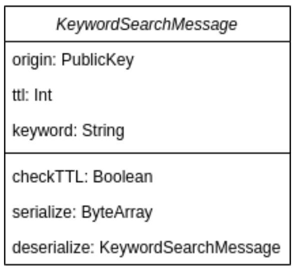


Figure 4.6: KeywordSearchMessage object sent over IPv8 in MusicCommunity

merely contains metadata of the tracks. We design the network to have a separate channel for downloading the track files. This is to enable fast discovery and searching of Releases, as Release objects have a small byte size.

4.6. 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. Using these signatures, 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 a centralized system, a single entity (such as a corporation) performs moderation of the authenticity of users and content. In a permissionless infrastructure, maintaining a record of legitimate parties is challenging. Reaching consensus is non-trivial. 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 is a simple implementation of the Trust on First Use (TOFU) model (Toth & Vlieg, 2013).

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

In order to minimize latency for discovering and playing music tracks, while using no central nor high-throughput servers, the network demands participants to upload content continuously.

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.

We design the app to, by default, seed the content of $\leq k$ Releases, which are sorted in descending order on publication date. This means that the seeding protocol uses a first-in-first-out strategy. This simple heuristic is used because of ease of implementation, because it requires no communication between devices. However, it could lead to unhealthy BitTorrent swarms for music released long ago. A peer-to-peer load balancing mechanism could prevent this but is out of scope for this thesis because of complexity.

Seeding of content should be done as a background process on the phone, so that it is still running when the app is not in use, and therefore maximize the availability of content. However, this is constrained by networking hardware, data subscription plans and other software running on the phone. The effectiveness of this approach should be evaluated with through experimentation.

4.10. Distributed search

For searching content we use introduce our simple distributed algorithm. Pseudocode of this algorithm is shown in alg. 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. 4.6.

This algorithm is inspired by the more general Gnutella search and retrieval protocol (Kronfol, 2002). Gnutella is widely used and has over a million users.

Fig. 4.7 shows a visualization of execution of the algorithm in a small network. Participant A wishes to find a set of results R for query q, so it initiates a search query. Firstly it inspects its local database. Because there are $|R| = 0$ results, it sends a query to its neighboring peers (B and C). The peers depicted in grey finds $|R|1$ results; they send their result back directly to A and do no other action. Other peers forward q to its neighbors, reducing q_{ttl} by 1. *Time-to-live* hits 1 when arriving in F,G,H so the search terminates.

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 $ttl \leq 0$ then
        return
    end if
    $ttl \leftarrow ttl - 1$
    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
```

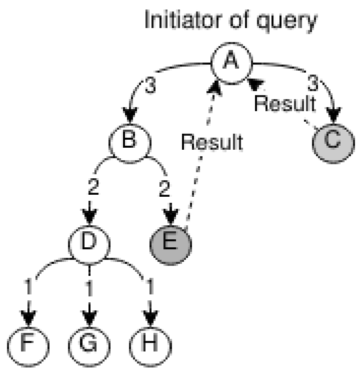


Figure 4.7: Search algorithm execution example: node A initiates a search query

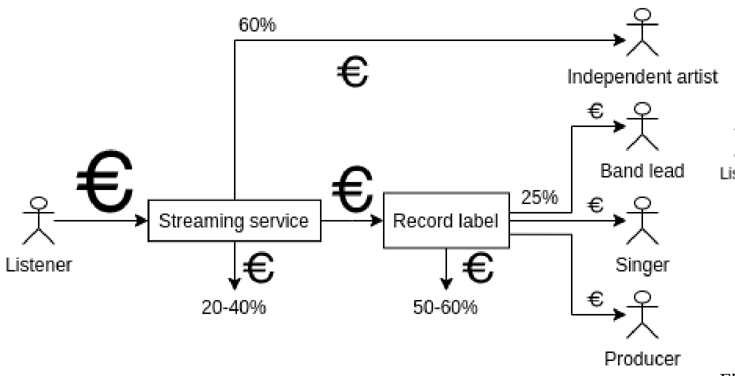


Figure 4.8: Money flow: current situation (simplified)

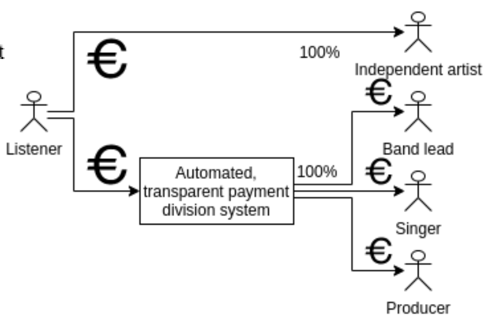


Figure 4.9: Money flow: desired situation

4.11. Transparent money flow

Figures 4.8 and 4.9 visualize the difference of money flow 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 4.12).

Fig. 4.9 shows another component: an automated and transparent payment division system. Currently, record holders have the task of paying artists, but the contracts and systems are often opaque and result in slow and low payouts. Automating this process should reduce these problems. 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.

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.

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 required to keep their private key safe.

4.12. 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.

The algorithm is flexible in the way of measuring interaction. It could be measured in e.g. time listened, plays, feedback in the form of likes or any linear combination of such. These parameters could be voted on by the community. The details of this division is shown in pseudocode, in alg. 2. We choose to only create transactions to the first k artists in order to have a constant message complexity for each iteration: $O(k) = O(1)$. This means that artists falling outside of this value of k do not receive money from the corresponding listener, so the value of k becomes a trade-off between network bandwidth usage to process the payments and fairness for artists.

Algorithm 2 Artist Income Division Algorithm (AIDA)

```
artistInteraction←Map(key: walletID, value: amount)
function SUBSCRIBE(artistInteraction, fee, k)  ▷ User registers for subscription  ▷ Artist interaction map:
    repeat Every time interval $t$
        artistInteraction←orderDesc(artistInteraction)    ▷ Order artist interaction descending
        topArtists←pickFirst(k, artistInteraction)        ▷ Pick the top k artists
        total←0
        for (walletID, value) in topArtists do
            total←total+value
        end for
        for (walletID, value) in topArtists do
            amount← $fee \cdot \frac{value}{total}$
            TRANSACT(amount, walletID)                    ▷ Peer-to-peer monetary transaction
        end for
    until unsubscription
end function
```

4.13. Content popularity algorithm

In order to present users with content that is reachable (downloadable), we track *content healthiness*. Content healthiness is the amount of peers that have and upload some specific piece of content. In the context of music this is the amount of uploaders for a music release (see fig. 4.5). This idea is heavily influenced by the *swarm health* terminology as part of BitTorrent, which interprets the number of seeders and leechers.

Our gossip-based algorithm used for spreading information about content healthiness is presented in alg. 3. Gossip-based algorithms are suitable in distributed systems because of their simplicity and flexibility (Kermarrec & Van Steen, 2007). Our algorithm is influenced by previous work present in the Tribler application (J. A. Pouwelse et al., 2008).

In essence, the algorithm works as follows: every peer in the network communicates with a few random neighboring peers periodically. The sending peer sends one message to each of its selected peers containing information about how well content (e.g. a musical Release) is available, measured by the amount of devices that have this piece of content. Every peer in the network has a local registration of content popularity, and updates this periodically. All messages are timestamped, so outdated information is removed after time.

Algorithm 3 Gossiping algorithm for content popularity

```
localDatabase ← Map(key: item, value: health)
function GOSSIP(localDatabase, peers, t, margin)
    repeat Every $t$ ms
        for healthItem in localDatabase do
            if timestamp ≤ timestampNow − margin then
                localDatabase->remove(healthItem)
            end if
        end for
        healthItem←pickRandom(localDatabase)     ▷ Pick a random item
        orderDesc(localDatabase)                 ▷ Order based on health
        healthItem2←pickFirst(localDatabase)
        peer←pickRandom(peers)
        sendGossipMessage(peer, list(healthItem, healthItem2))
    until end
end function
function ONGOSSIPMESSAGE(items)                  ▷
    localDatabase->overwrite(item)
end function
```

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 trustless 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. MusicDAO runs on Android 5.1.1 and above, meaning there are 13,734 Android devices supported[1]. The main features of our work in comparison to the state-of-the-art related work (see sec. 3.3) can be viewed in table 5.1.

Features overview

- Defining digital musical releases using metadata, and sharing those with peers;
- Immutable storage of music metadata and cryptographic identification of artists;
- Peer-to-peer music streaming over BitTorrent; Prioritization algorithm to minimize streaming latency; Caching of audio files and metadata.
- Browsing playlists; Local and remote keyword search for music.
- Mobile Bitcoin wallet implementation; Peer-to-peer donations to artists using Bitcoin.
- Content popularity gossip-based algorithm; Content sorting algorithm based on content popularity.

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 5.2. In the following sections we explain the details of implementing the aforementioned features in a top-down overview, starting with the interfaces.

The Artist Income Division Algorithm, presented in the Design chapter, is not implemented in MusicDAO at the time of writing. The implementation does contain the central component of this algorithm: an integration with peer-to-peer direct donations without intermediaries. For this reason, AIDA could be implemented trivially by another reasearcher or developer.

[1]According to Google Play Console, 17-02-2021.

	>99% to artists	Stable currency	Governance system	Fully P2P and self-scaling
Audius	×	×	✓	×
Musicoin	×	×	×	×
Opus Audio	×	×	✓	×
This thesis	✓	×	×	✓

Table 5.1: Overview of our work compared to related work

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 5.2: Notable libraries used

5.1. Zero-server infrastructure Android application

The system described in chap. 4 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.

MusicDAO is an Android application which runs on any Android device running version 5.1.1 or above. It 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[2] and runs on Android 5.1 and above. Its code is publicly available[3]. As programming language Kotlin is selected, as it is the preferred language for Android development[4]. 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 5.9). An overview of the structure of the code can be seen in the package diagram in fig. 5.11.

5.2. 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 fig. 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 fig. 4.5). The playlists are rendered in real-time based on TrustChain data. This means: during browsing, newly clickable playlists show up near-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.

The cover art shown in fig. 5.1 is rendered using MP3 metadata processing, using its ID3 tags (when available). The system does not contain a pre-fetching mechanism for obtaining cover art (as observed in fig. 5.5). Instead, it is only shown after at least one of the MP3 metadata has finished transferring using the BitTorrent stream.

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

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

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

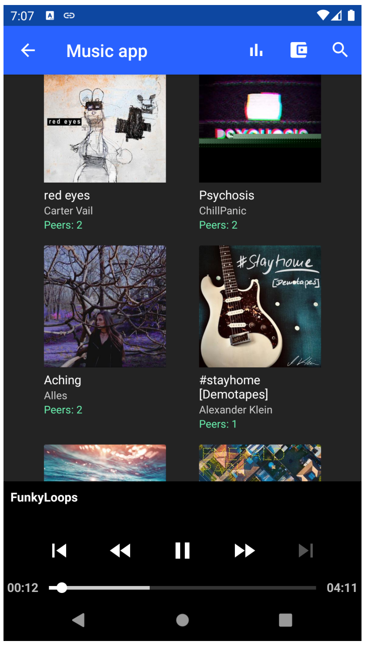


Figure 5.1: The playlist overview screen, which is the entrance screen

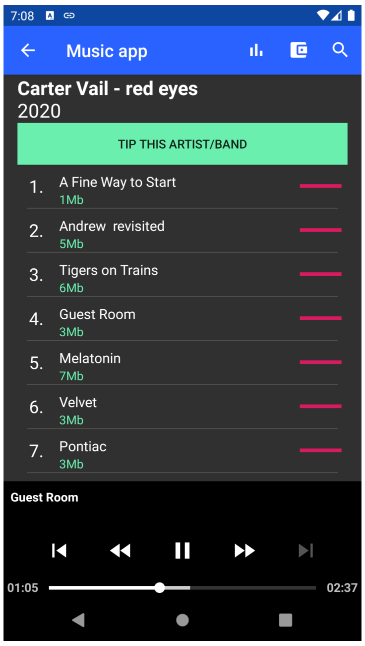


Figure 5.2: Playlist fragment, showing all tracks of one Release, streaming a track

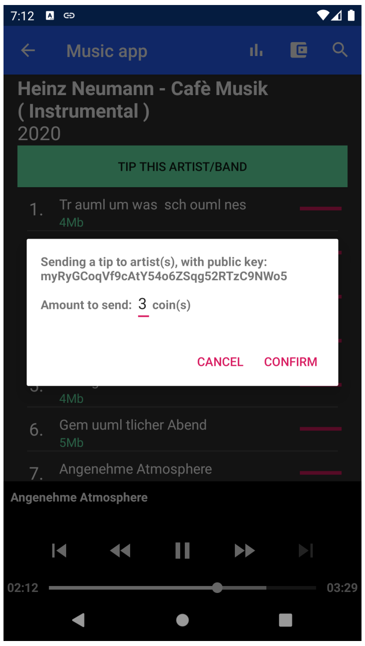

7:12
Music app
Heinz Neumann - Cafè Musik (Instrumental)
2020
TIP THIS ARTIST/BAND
1. Tr auml um was sch ouml nes
4Mb
Sending a tip to artist(s), with public key:
myRyGCoqVf9cAtY54o6ZSqg52RTzC9NWo5
Amount to send: 3 coin(s)
CANCEL
CONFIRM
4Mb
6. Gem uuml tlicher Abend
5Mb
7. Angenehme Atmosphere
Angenehme Atmosphere
02:12
03:29

Figure 5.3: Sending a tip to an artist or band

5.2.1. Distributed keyword search algorithm

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 alg. 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 4.6) 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 the network we use a time-to-live property.

Algorithm 1 is implemented using the parameters $ttl = 1$ and $maxPeers = 20$ so there is no recursive searching throughout the network. A low value of ttl is chosen because any $ttl > 1$ results in an exponential growth of search messages throughout the network.

5.2.2. Gossip-based algorithm for content popularity

The gossip-based algorithm used for communicating content popularity with peers is implemented as described in alg. 3. Every peer in the network sends a content health message to a random peer every 3 seconds. The content healthiness is displayed for every release as *Peers: X* (see fig. 5.1). Gossip messages older than one hour were discarded ($margin = 3600$). It is non-trivial to determine a good value for $margin$ as it is a trade-off between amount of information and recentness of information.

5.3. Music streaming algorithm

To achieve fast buffering of music, we implemented a priority system for tracks and for parts of tracks (chunks) (see 5.6). In essence, the player prioritizes chunks that the user is currently interested in, by actively asking peers to send the corresponding k 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 (see libtorrent Manual[5]). 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 (5.2). 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

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

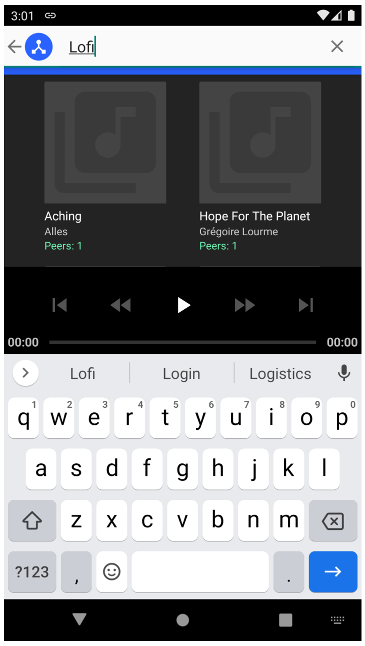


Figure 5.4: Entering a search keyword

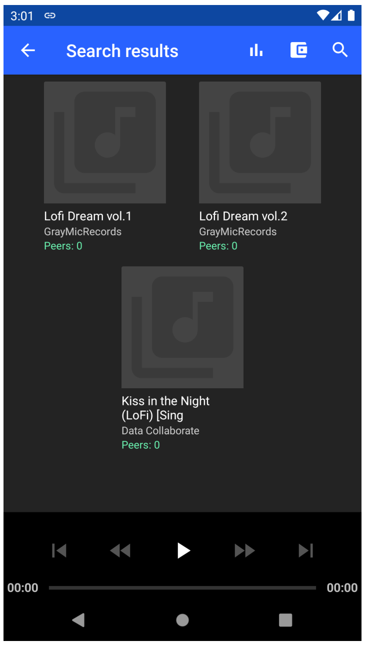


Figure 5.5: Search results from peers, using keyword filtering

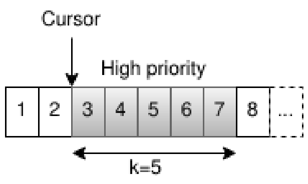
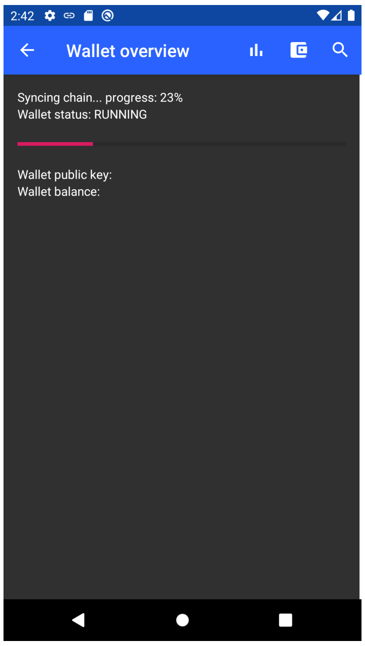


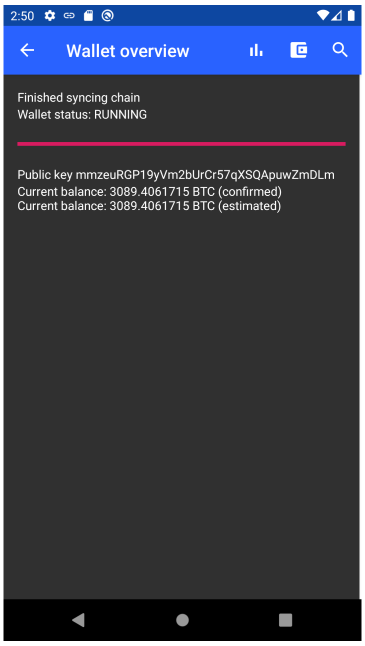
Figure 5.6: Music streaming mechanism



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

Wallet public key:
Wallet balance:

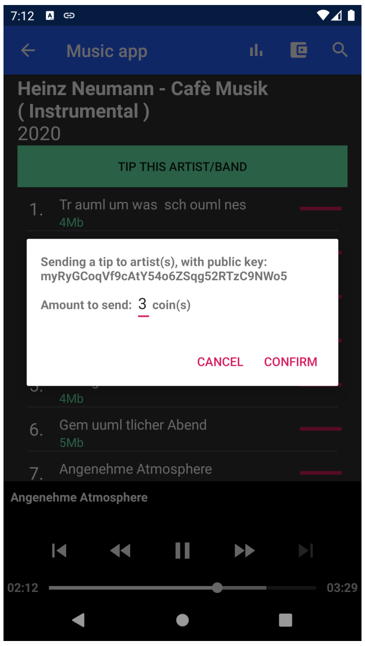
Figure 5.7: Synchronizing with the Bitcoin RegTest environment blockchain



Finished syncing chain
Wallet status: RUNNING

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

Figure 5.8: Wallet overview and balance after synchronizing



Sending a tip to artist(s), with public key:
myRyGCoqVf9cAtY54o6ZSqg52RTzC9NWo5

Amount to send: 3 coin(s)

Figure 5.9: Sending a tip to an artist or band

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.

5.4. Peer-to-peer financial infrastructure

Our system contains peer-to-peer payments where 100% of money goes to artists. We created a public Bitcoin RegTest environment[6] 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 participating in the MusicCommunity is given a private/public wallet identity upon installation of the MusicDAO. The wallet interface (fig. 5.7) shows synchronization status of the RegTest network (see 5.4). Once the wallet is fully synchronized with the blockchain, the private key and balance are displayed as shown in fig. 5.8.

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. In a real-world situation, bitcoin nodes should be found by querying peers for bitcoin node addresses.

Upon browsing a playlist, a donation button is displayed as shown in fig. 5.9. 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. After confirmation, the transaction is registered on the RegTest network (see 5.4). In the current software implementation, when a user makes a donation, this transaction is atomic and can only go to one wallet. An automatic splitting system between different artists of one group or band has not been implemented yet.

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

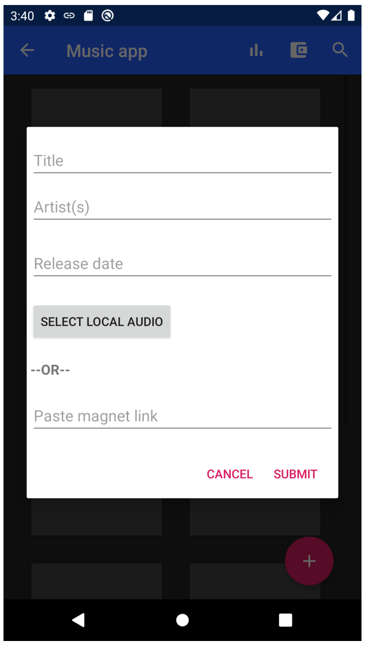


Figure 5.10: Dialog for creating and publishing a new Release

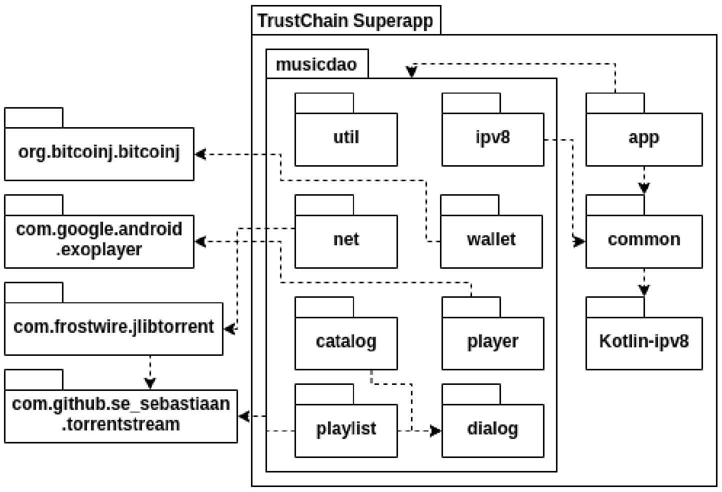


Figure 5.11: Package diagram showing the interaction with external libraries and TrustChain Superapp packages

5.5. Permissionless publishing of music

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). 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.

To create and share music content, a user can create a Release object using the dialog shown in fig. 5.10. 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 table (*Distributed Hash Table*, 1981-2019). In addition, the app uses the local peer discovery (LPD) (8472, 2015) functionality of BitTorrent. This allows for finding peers and transmitting torrent pieces over a LAN, resulting in fast transmission and low latency. 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.

5.6. 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.

[7]https://github.com/frostwire/frostwire-jlibtorrent

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 5.3: Code coverage overview

5.7. Quality assurance

As mentioned in the introduction, the goal of the implemented work is to be the first steps towards a full robot economy. Aside from being a functional real-world application, MusicDAO also presents an DAO AI framework to create other robot economy applications, beyond the scope of music. Experimentation in other domains allows to gain a further understanding of the developing science of the robot economy.

As the framework presented in MusicDAO is an important milestone towards the robot economy, the code reusability is of great importance. Therefore we used best-practice code quality assurance approaches such as continuous integration, code linting and pull request reviews. These best practices are widely used in the software industry.

We use a continuous integration (CI) environment, run by Github Actions[8]. GitHub Actions is a mature CI system, used by e.g. the Google Cloud Platform. Within our CI environment, we use automated tests and code linting. 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 5.3. Code linting is performed using KtLint[11].

Code coverage is measured by Android Studio 4[12]. The majority of uncovered code contain user interface interaction and networking callback logic. Code coverage could have been improved by introducing Android instrumented tests. These are tests that run on an Android device or emulator so that user interaction flows can be tested. However, we chose to not implement this as these type of tests can not be run in our continuous integration environment, and tweaking the CI for support of this would be a non-trivial task.

MusicDAO runs on Android 5.1.1 and above, and as such supports 13,734 different Android devices[13]. During the implementation phase we ensure that it runs well on different screen sizes and Android versions, by running it on 20 different Android devices. To gain a further understanding of device compatibility and bug hunting, we make use of an online crash reporter (fig. 5.12). Firebase Crashlytics[14] registers app installs and crashes in real-time, which we use during the experimentation of MusicDAO. It shows details of each application crash for all users (see fig. 5.12). Using this, a full stack trace can be inspected remotely for each crash. In addition it helps with gaining insight in various performance metrics for certain types of Android devices.

[8]https://github.com/features/actions
[9]https://junit.org/junit4/
[10]https://mockk.io/
[11]https://github.com/pinterest/ktlint
[12]https://developer.android.com/studio
[13]According to Google Play Console, 17-02-2021.
[14]https://firebase.google.com/docs/crashlytics

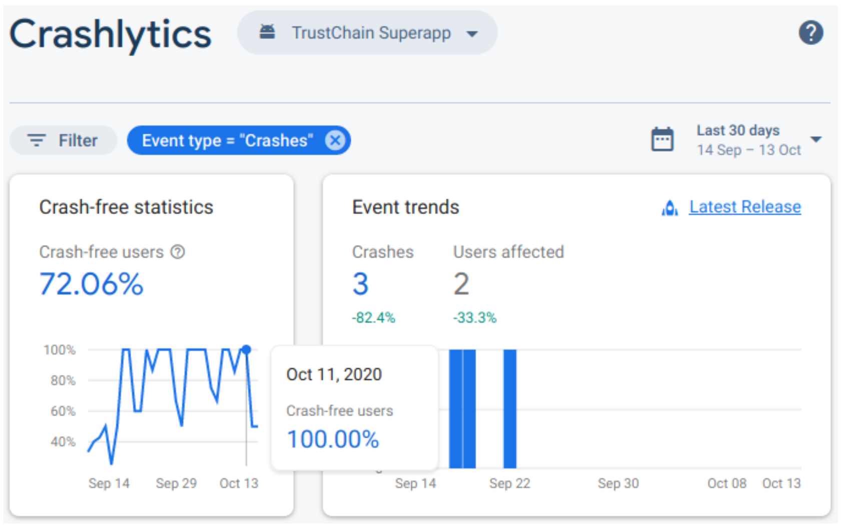


Figure 5.12: Firebase Crashlytics crash reporter; this figure shows the crashes over time

6
Evaluation

Our system was installed and executed on 50+ Android devices in a public release. Its crash-free rate throughout this public experiment was > 94% as measured by Crashlytics[1] (see fig. 6.1). 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. 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] `https://firebase.google.com/docs/crashlytics`

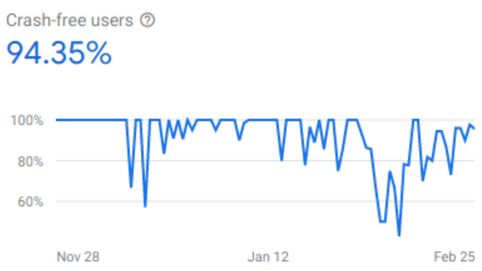


Figure 6.1: Crash-free users over time, as measured by Firebase Crashlytics

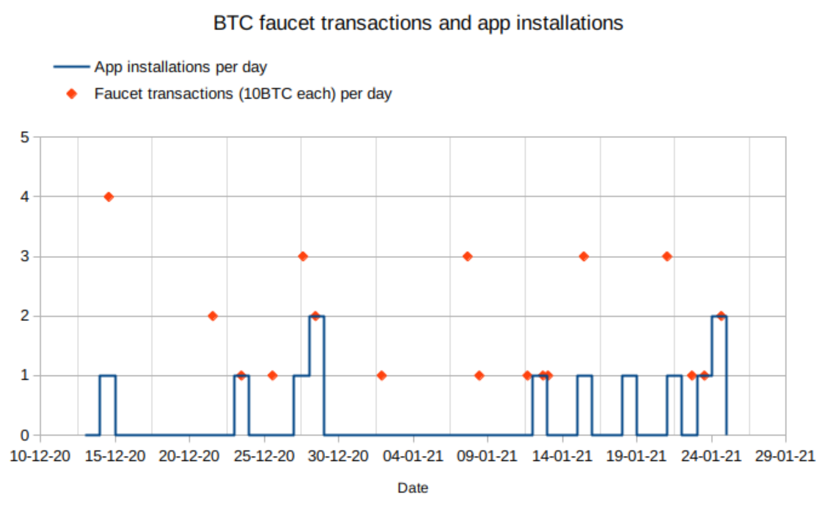


Figure 6.2: Graph showing the relationship between faucet transactions of 10BTC and app installations. Every app installation should correspond to at least one faucet transaction.

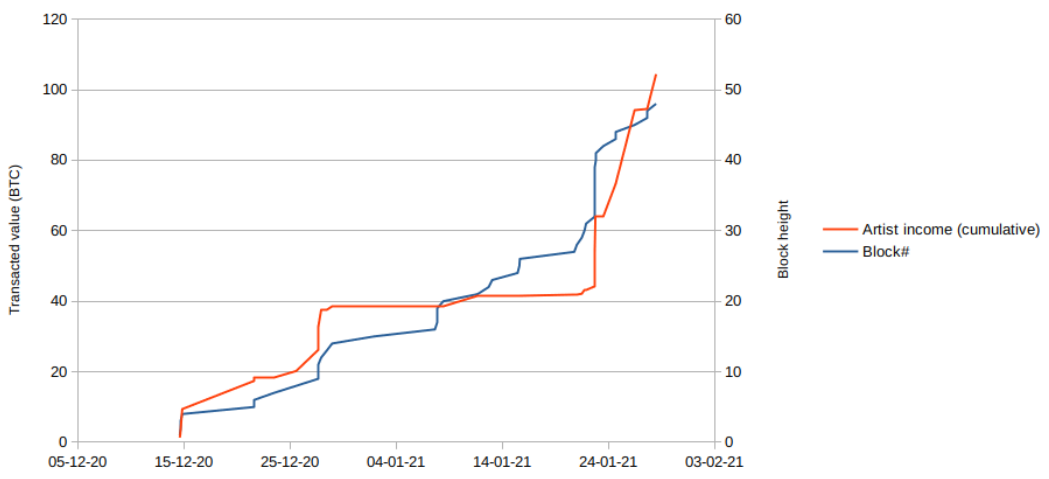


Figure 6.3: Artist income (cumulative) over time and block creation

1. App installations per day, as measured by the Google Play console[2].
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 in fig this is not always the case. On 19 Jan 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 console[3]) or app installations outside of the Play store.

6.1.2. Donation money flow

The datasets used to create the plots shown in figs. 6.3 and 6.4 are received by scraping blockchain block data from our Bitcoin node. Fig. 6.3 shows the relation between created blocks and money transacted per block (cumulative).

We observe that artist income is 104.4 coins over a time-span of 45 days. We asked a small group of 11 people to install the app and spend all their starter money (10 coins) on donations to artists. During this

[2]`https://play.google.com/console`
[3]`https://play.google.com/console`

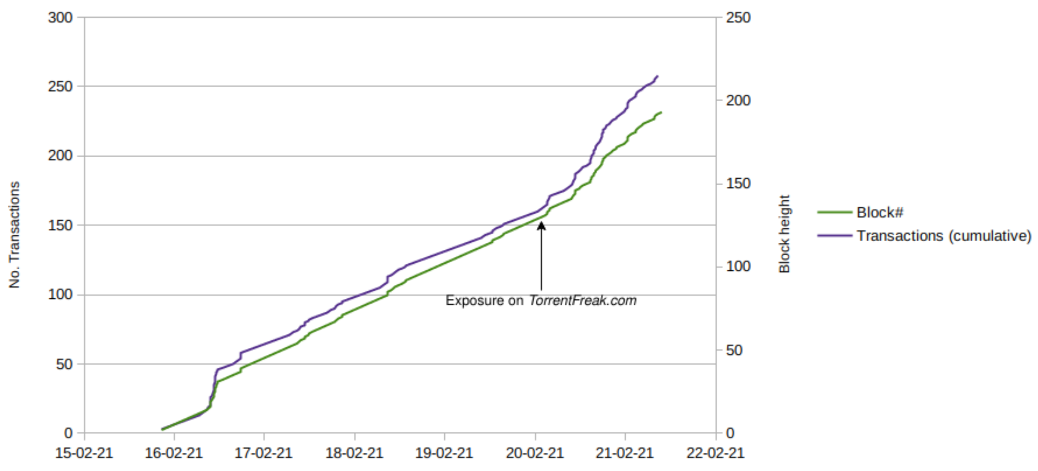


Figure 6.4: Transactions (cumulative) over time and block creation (1 hour block interval)

time-frame, there were at least 12 unique app installations through the Google Play console, which means that these users were given a combined budget of at least 120 coins. This conforms the validity of the artist income measurement and shows the effectiveness of the decentralized financial infrastructure in a small network of users.

During a 7-day public trial using an online news publication for user growth, we configured the Bitcoin node to mine one block per hour. This block creation interval is well adopted by the node, as a steady linear growth can be seen in fig. 6.4. This graph also shows the effect of an influx of users: On 20 Feb 2021, MusicDAO was promoted in a news article[4]) leading to a 220%+ growth in app installations on a weekly basis. In the few days after this moment, the average transactions per block was significantly higher.

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).

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

6.5 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. 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

[4]`https://torrentfreak.com/university-runs-massive-bittorrent-seedbox-to-showcase-music-streaming-app-210220/`

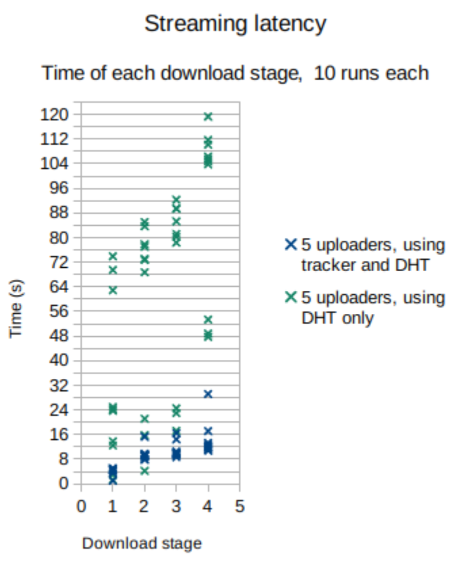


Figure 6.5: Average time spent per download stage. Measured by 20 runs in total, in two different network configurations.

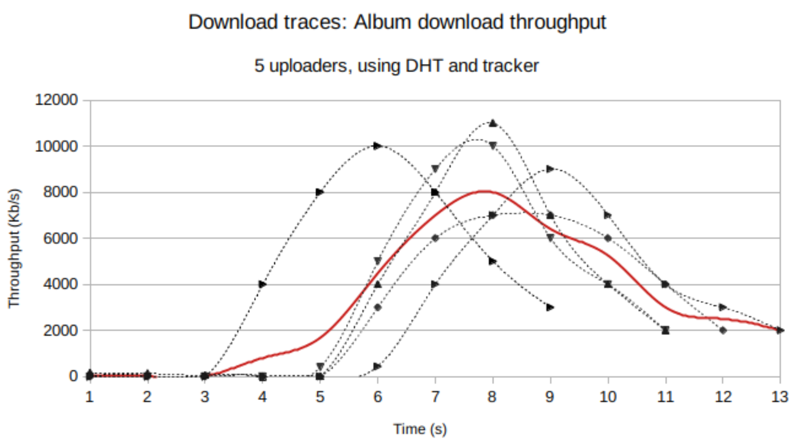


Figure 6.6: 5 traces of download throughput, downloading an album of around 38 Mbs. Measured with Nokia 7.

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 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).

6.6 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 6.5).

The datasets used to create the plots shown in figs. 6.3 and 6.4 are received by scraping blockchain block data from our Bitcoin node. Fig. 6.3 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.

Note that all devices evaluated in experiments 6.5 and 6.6 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 over TCP across NATs.

6.2.2. File headers and BitTorrent piece ordering

The streaming algorithm implemented and evaluated uses the BitTorrent priorities system. The first $x = 5$ pieces at the selected portion of the file (by default, the start of the file) have a higher priority than the other pieces. However, even though some pieces A have higher priority than other pieces B, BitTorrent does not guarantee that all $a \in A$ are received earlier before any $b \in B$. Therefore, the actual ordering of the received pieces is non-deterministic. The simple heuristic used in the streaming algorithm is: start playing the file when 30% has been downloaded; but this does not always work and this leads to inconsistent stream buffer times. Additionally, we use the assumption that the header of the audio file is at the first few bytes of the file, but this is not always true.

There is significant ongoing research related to peer-to-peer streaming in literature that can be used to improve our streaming algorithm (Erman, Vogeleer, & Popescu, 2008) (Akkanen, 2017).

6.2.3. Content discovery

Fig. 6.7 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 content discovery over time, we compare three different experiments with a mathematical model for expected discovered items over time.

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

Every time interval, all devices send one music block to a random neighbor. As such, the expected amount of blocks received in one interval is 1. When receiving a music block, the chance that it is a block that has not been received before is

$$\frac{R-r}{R}$$

with $r \leq R$ the amount of items received so far. It follows that the expected value for the cumulative amount of unique items discovered after x iterations is

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

where

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

This expected value $E[R]$ over time is plotted in 6.7 as Model.

As every device sends one item per time interval to a neighbor, the gossiped items per time interval is equal to network size n. This means message complexity M is equal to network size:

$$M = O(n)$$

Every device participating in gossiping sends 1 message and receives, on average, 1 message back. So the expected messages to process is $E[M] = 2$ per iteration. Therefore the message complexity for a single device is

$$M = O(1)$$

This means that the network size does not affect the message processing workload of each device, and makes the algorithm highly scalable.

In 6.7 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.

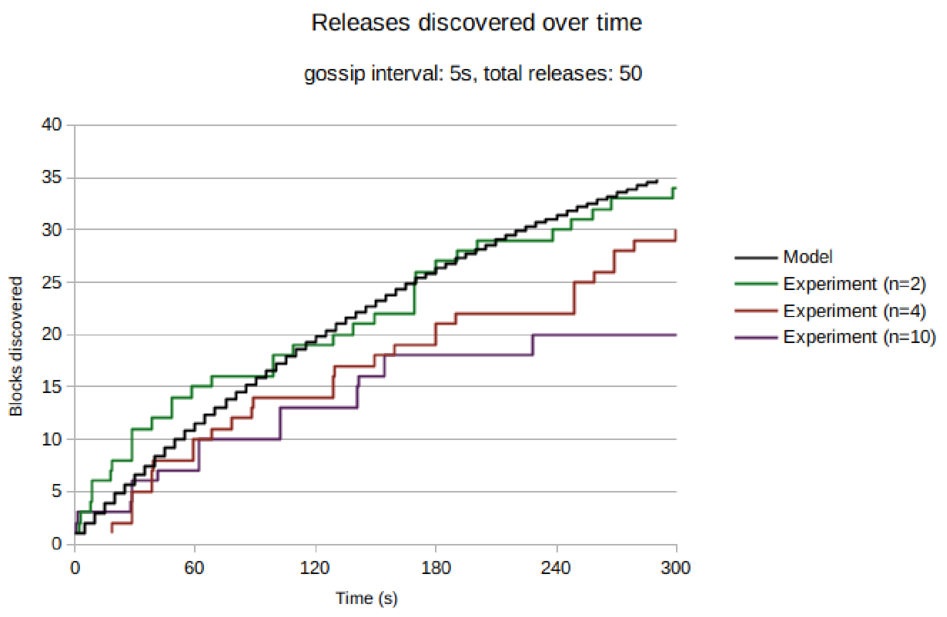


Figure 6.7: Measurements of content discovery: releases discovered after a fresh installation. n: network size (amount of Android devices)

6.2.4. Random access latency

Random access latency of metadata 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 distributed search algorithm (alg. 1) is used with the following parameters.

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

This means that a maximum of 20 neighbors are contacted when performing a search, and that search is not recursive; meaning only neighbors are contacted (not neighbors of neighbors). We expect that random access latency does not increase in correspondence to relation to its network size.

For three different network sizes of 2, 6 and 10, 10 runs are done and the latency and average latency are plotted in 6.8. 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.

It should be noted that message complexity of the search algorithm grows exponentially with the value of *time-to-live* (ttl). For time-to-live T and the maximum number of neighboring peers to ask P, the amount of messages for a single search is p^T for $p \leq P$. This leaves a message complexity of $O(P^T)$. This could be improved by using a different distributed data structure that is optimized for search, such as a balance tree or a distributed hash table.

6.3. Device connectivity and network bootstrapping

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.

Our goal, as described in the Design section, was to create a phone-only network. This is not fully accomplished as we use a bootstrap node to tackle the *peer-to-peer bootstrap problem*. As recognized by Wolinsky, Juste, Boykin, and Figueiredo (2010), the peer-to-peer bootstrap problem is the problem of finding peers in a

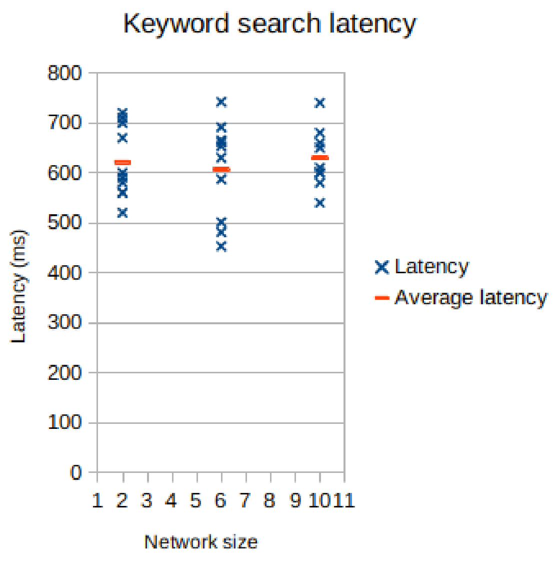


Figure 6.8: Search latency of performing keyword search

P2P overlay network, when there are no connected peers to begin with. This is a widely known, yet unsolved problem for any peer-to-peer network, and it is also relevant in MusicDAO.

It must be noted that a bootstrap node is not necessary when you can connect to a peer that is already on the MusicDAO network, for example via a local area network or Bluetooth. Moreover, a bootstrap node can be discarded once the bootstrapping procedure is successful.

6.4. Scalability

We presented an infrastructureless network of phones that together form a fully operating application. During supervised experiments, a network size of up to 10 phones was used. Now we present our expectations of scalability beyond 10 active devices.

As every device cooperates in the network by being both an uploader and downloader for content, the network size should not affect download speeds for users. The bandwidth, processing power and storage capacity of the application should naturally grow with the amount of users. An important note is that we do not take into account free-riders (peers that use the network but do not deliver resources) or adversaries (peers that perform attacks such as network flooding or sybil attack).

The system 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 (in contrary to centralized streaming infrastructure).

The transaction system, Bitcoin, may become a bottleneck in scalability (Chauhan, Malviya, Verma, & Mor, 2018). It has a single, global blockchain and blocks take on average 10 minutes to mine. There are other, more scalable protocols for transacting cryptocurrency in active development (LeMahieu, 2018), but they are not yet mature. However, our financial system of wallets using public-key cryptography is easily adaptable to any other cryptocurrency. We do not consider centralized technologies as that would concentrate power, in contrary to the aim of this thesis.

6.5. Bitcoin node and transaction fees

We made use of a bitcoin miner, running on a dedicated server continuously throughout the experimentation phase. At least one miner is necessary on our test network in order to process payments. The use of a dedicated server contradicts our design of a zero-server software system, and creates a single point of failure. However, this situation could be improved by decentralizing the network: by running miners on each device participating in the network, the resources become more balanced, and this would reduce the risk of hardware failures.

We use Bitcoin as peer-to-peer electronic cash system. Bitcoin uses transaction fees for every transaction. In our unsupervised experiments with MusicDAO, we observed an average transaction fee of 0.00003778 coins. As such we cannot say that artists receive 100% of the donation money from users, but rather roughly 99.9996%. This fee is calculated dynamically. It is based on the size of a transaction, and network conditions such as congestion.

Our framework for the robot economy does not explicitly state the type of currency used. In MusicDAO, Bitcoin can be trivially swapped for any other currency as long as that currency uses a digital peer-to-peer public key infrastructure with public wallet addresses. As such, a different crypto-currency with lower transaction fees (such as Nano (LeMahieu, 2018)) can be used.

6.6. Battery usage

The challenge of any peer-to-peer mobile app is battery usage, as it should be connected and contribute to the network as much as possible to optimally support other peers. To do this, TrustChain Superapp sends regular keep-alive messages in order to track the status of peers. To do this, the Superapp runs in the background as an Android service. Therefore it can still be used when the app is closed. The downside is that this appears to have a major effect on battery usage, as found in previous experimentation by Skala (2020). He found that, during a 10 hour experiment, when a Google Pixel phone runs the app in the background and is connected to 20 peers, its battery consumption is roughly 10 times higher than normally.

Battery usage could be improved by adjusting the peer-to-peer connectivity protocol of IPv8, such as by reducing the interval of keepalive messages. This requires further investigation and more research.

7 Conclusion and Future Work

This thesis presents a novel framework for building a robot economy in software. This framework allows for designing and implementing software for the common good: software that (1) handles financial transactions in a fair way (2) as decided by democratic engagement, (3) runs transparently and autonomously, (4) is open to any participant (permissionless), (5) is decentralized and leaderless, (6) supports a self-evolving codebase, and finally (7) can make intelligent decisions on its own using AI.

This thesis presents a music streaming alternative for the Big Tech industry: we implemented our framework in order to build a fully working decentralized music streaming, publishing and discovery mobile app with peer-to-peer donations. In this proof-of-concept called MusicDAO, we applied the theory of the robot economy in software to an industry of which the core contributors (artists) suffer majorly from centralization on the Internet: the music industry. Most music streaming platforms take a 20-40% cut of music revenue; MusicDAO takes 0%. MusicDAO implements 1, 3, 4 and 5 of the key features of a robot economy in software (see above).

MusicDAO is completely free of label and platform intermediaries taking large revenue cuts. It forwards > 99.99% of all music revenue to artists. All money flow is public and transparent. Artists using our DAO can be independent and self-publishing. Apart from the *music platform oligarchy*, it also surpasses the *label oligarchy* as it requires no music label to publish music.

MusicDAO implements a peer-to-peer audio streaming algorithm, which is able to stream any BitTorrent audio swarm. Its buffer loading time does not reach industry standards yet but this can be improved by more optimized pre-fetching and piece ordering algorithms.

Our system is not yet resilient against free-riding, spam attacks or sybil attacks. Preventing such attacks is an active developing area in distributed systems research.

7.1. Generality: beyond music

The framework presented in this thesis can be applied to many other domains beyond music. Most functions of corporation-run centralized platforms are already automated, and handled by AI or robots. It can transform Internet platforms, or even complete value chains that currently have unfair or opaque money flows. One may think of a production and sales chain with no retail cut, or a video publishing network with a subscription model without intermediaries. This thesis aims to be a step into an evolving research direction, *infrastructure for the common good*: autonomous software that works in favor of its user community instead of companies or profit.

7.2. Future research directions

In the MusicDAO proof-of-concept, there is no use of intelligent robots. Decision making based on AI is the next key step to the robot economy in software. We envision a symbiotic interaction between humans and robots: humans vote for the baselines and rules within which robots play, while robots use AI such as machine learning or evolutionary algorithms to perform tasks and make decisions.

Content moderation is a key ingredient of any Internet platform. In MusicDAO, any music that is published is automatically admitted to the network. Therefore the network does not automatically remove any duplicates (copies), illegal remixes or other illegal audio. Using DAO and intelligent robots, a system can be created in which robots moderate content using AI, while humans democratically vote on the rules for content.

Ongoing research into Self-Sovereign Identity (SSI) can fix part of this problem: SSI can create a certain passport for content creators, with which creators can prove their identity and thus ownership of certain content.

Streaming income is only a part of revenue from music. MusicDAO can be extended to include other revenue flows to artists, to make also those incomes higher and more transparent for artists. Possibilities are e.g. live concert tickets, merchanidise or physical audio sales.

A robot economy is highly susceptible for the problem of *determining responsibility* when things go wrong. For example, a software bug may occur or a robot can make an unexpected monetary transaction. This challenge requires research into new theories and ideas in the context of law and philosophy for the 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.

Akkanen, J. (2017, September 19). *Continuous scheduling for peer-to-peer streaming.* Google Patents. (US Patent 9,769,255)

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

Arduengo, M., & Sentis, L. (2020). The robot economy: Here it comes. *International Journal of Social Robotics*, 1–11.

Bazinet, J. B., Singlehurst, T., May, M., Suva, J., Ezawa, K., & Yap, A. (2018). Putting the band back together: Remastering the world of music (issue brief). *Citi GPS.*

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.*

Benet, J., & Greco, N. (2018). Filecoin: A decentralized storage network. *Protoc. Labs*, 1–36.

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).

Chauhan, A., Malviya, O. P., Verma, M., & Mor, T. S. (2018). Blockchain and scalability. In *2018 ieee international conference on software quality, reliability and security companion (qrs-c)* (pp. 122–128).

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

Das, T. (2016). Intelligent techniques in decision making: a survey. *Indian Journal of Science and Technology*, *9*(12), 1–6.

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

Erman, D., Vogeleer, K. d., & Popescu, A. (2008). On piece selection for streaming bittorrent. In *Fifth swedish national computer networking worshop (sncnw2008).*

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).

Friedlander, J. P. (2020). Riaa 2019 year-end music industry revenue report. *Recording Industry Association of America. https://www.riaa.com/reports/riaa-releases-2019-year-end-music-industry-revenue-report/.*

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.

Jaspers Focks, T., van Veltom, H., Wigmore, V., & Nguyen, W. (2018). Plebnet: A self-replicating botnet for privacy protection.

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).

Kermarrec, A.-M., & Van Steen, M. (2007). Gossiping in distributed systems. *ACM SIGOPS operating systems review, 41*(5), 2–7.

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

Kronfol, A. Z. (2002). *Fasd: A fault-tolerant, adaptive scalable distributed search engine* (Unpublished doctoral dissertation). Master’s Thesis http://www. cs. princeton. edu/akronfol/fasd.

LeMahieu, C. (2018). Nano: A feeless distributed cryptocurrency network. *Nano [Online resource]. URL: https://nano. org/en/whitepaper (date of access: 24.03. 2018).*

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`

Music, R. (2015). Fair music: Transparency and payment flows in the music industry. *Rethink Music.*

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

Pelly, L. (2017). *The problem with muzak.* Retrieved 09 November 2020, from `https://thebaffler.com/salvos/the-problem-with-muzak-pelly`

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.*

Toth, G. X., & Vlieg, T. (2013). *Public key pinning for tls using a trust on first use model* (Tech. Rep.). Technical report, University of Amsterdam.

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/`

Wolinsky, D. I., Juste, P. S., Boykin, P. O., & Figueiredo, R. (2010). Addressing the p2p bootstrap problem for small overlay networks. In *2010 ieee tenth international conference on peer-to-peer computing (p2p)* (pp. 1–10).