Master's thesis

NTNU
Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Computer Science

Joakim Grønstad Lindgren
Thomas Åge Langelo Røyset

A Study of Hybrid Architectures in the Realm of Media Streaming

Master's thesis in Master of Science in Informatics
Supervisor: Svein Erik Bratsberg
May 2019

NTNU
Kunnskap for en bedre verden

Joakim Grønstad Lindgren
Thomas Åge Langelo Røyset

A Study of Hybrid Architectures in the Realm of Media Streaming

Master's thesis in Master of Science in Informatics
Supervisor: Svein Erik Bratsberg
May 2019

Norwegian University of Science and Technology
Faculty of Information Technology and Electrical Engineering
Department of Computer Science

Abstract

Hybrid architecture is a distributed software architecture that has been used for media streaming in the past, but many media streaming services has since stopped using it. However, with several years worth of technological advancements, and the knowledge that media streaming uses a large portion of overall internet traffic, the question of hybrid architecture's viability for media streaming services should be discussed anew. Therefore, this thesis will take a look at services that use hybrid architecture, and find advantages and disadvantages, which are analyzed to determine properties that are prominent in such services. This will be used to determine the viability of hybrid architecture for media streaming services.

The research is a case study in which hybrid architecture is the environment, with media streaming services as the instance. Services are selected based on usage, history and what they provide of content. This is in order to cover a wide area of hybrid architecture and media streaming. The selected services serves as a basis for finding properties. Based on the document analysis of literature on these services, five specific properties were found to be recurring. When discussing the use of these properties in media streaming, only three out of five properties were found to be significant. The two remaining properties were deemed to be too general, albeit still important.

To determine the viability, the properties were analyzed, and selected, non-hybrid media streaming services were used as comparison. The results showed that although there are disadvantages to using a hybrid architecture, it is viable in relation to media streaming. However, there might be better alternatives, and further research is required on both a theoretical and an experimental level, to increase the utility of hybrid architecture.

Sammendrag

Hybridarkitektur er en distribuert programvarearkitektur som tidligere har blitt brukt av strømmetjenester. Mange av disse tjenestene har siden da sluttet å bruke denne arkitekturen. Takket være store teknologiske fremskritt, og kunnskapen om at strømmetjenester står for en stor del av den globale internettrafikken, burde hybridarkitekturens nytteverdi for strømmetjenester bli diskutert på nytt. Derfor vil denne oppgaven se på tjenester som bruker hybridarkitektur for å finne fordeler og ulemper. Disse fordelene og ulempene kan analyseres for å fastslå attributter som er fremtredende i slike tjenester. Disse attributtene vil bli brukt til å bestemme nytteverdien av hybridarkitektur for strømmetjenester.

Forskningen utført i denne oppgaven er en kasusstudie, hvor hybridarkitektur er utgangspunktet, med strømmetjenester som fokus. Strømmetjenester blir valgt basert på hvor mye de er brukt, historisk utvikling og hva tjenestene leverer av innhold. Valget ble gjort for å dekke et bredt område av hybridarkitektur og strømmetjenester. De valgte tjenestene brukes som et grunnlag for å finne attributter. Basert på dokumentanalysen av litteratur om disse tjenestene, identifiserte forskningen fem distinkte attributter. Under diskusjonen om bruken av attributtene i strømmetjenester, var tre attributter framtredende. De to gjenværende attributtene ble ansett for å være for generelle.

For å bestemme nytteverdien av hybridarkitektur, ble attributtene analysert. Utvalgte strømmetjenester, som ikke bruker hybridarkitektur, ble brukt som sammenligningsgrunnlag. Sluttresultatet viste at selv om det er ulemper ved bruk av en hybridarkitektur, har den en nytteverdi sett i sammenheng med strømmetjenester. Derimot kan det være bedre alternativer, og det kreves videre forskning på både et teoretisk og eksperimentelt nivå for å gjøre hybridarkitektur mer brukbart.

Preface

This master's thesis in Informatics was authored by two students, and submitted to the Department of Computer Science (IDI) at the Norwegian University of Science and Technology (NTNU).

We would like to extend our thanks and gratitude to our supervisor Svein Erik Bratsberg for providing sterling guidance in relation to the work on this thesis. We would also like to thank the people around us, who have provided useful suggestions or lent their ears during our work.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Media streaming has experienced a rise in popularity the last two decades. How media streaming services are designed has changed over the years to fit with demand, usage and technological advancements. This again means that the architecture of a media streaming service has changed to solve problems that have appeared. However, with the increase in user base, larger file sizes and better access to services, the need to address potential problems with bottlenecks and cost-efficiency are present. Hybrid architecture is a type of architecture that has been utilized in media streaming in the past, but for different reasons services transitioned away from it. Some of the potential problems found in hybrid architecture could be handled by technological advancements, thereby making it a possible solution for media streaming services. Therefore, this thesis will look at media streaming services and hybrid architecture, to determine if hybrid architecture is a viable option for media streaming services.

1.1 Background & Motivation

Media streaming has exploded in popularity, which can be shown by video streaming representing approximately three quarters of overall internet traffic in 2017[1]. Services such as YouTube and Netflix has become popular solutions in order to consume not only high production value content, but also user-created content. These services have grown the past few years, and are predicted to keep growing at a steady rate for the future. YouTube has been growing by, and is predicted to keep growing by, about 100 million users each year[2]. Netflix, on the other hand, has been growing with about 20 million users each year[3].

In 2017, YouTube reported that a billion hours of content is watched each day[4]. Using the information provided on the WhistleOut website on YouTube's data usage, and making the naive assumption that the content is in 480p resolution, that amounts to approximately 240 petabytes of data watched per day, 10 petabytes per hour, or 166.7 terabytes per minute[5]. The fact that the service is able to deliver that quantity of content, without users experiencing stuttering or the service crashing, is an amazing feat of human ingenuity. However, this feat is presumably costly in regards to how expensive it is to maintain server farms for hosting, as well as content delivery.

With technology improving at a fast rate, consumers will get better connections and devices every year. This can be seen in the report from Speedtest by Ookla, which stated that the worlds average internet speed increased with over 30% in 2017[6]. Furthermore, Moore's law had set expectations for devices to become faster, better and cheaper in the future[7]. Due to these properties, it is probable that consumers will request content with a higher definition than before. If the previous statement holds true, it will inevitably lead to increased file size and bandwidth usage as can be seen in Table 1.1. With new technologies being developed to enhance content experience, for example virtual reality, it will probably increase file sizes further[8]. Even with modern compression techniques and improvements in transcoding, there are still limits to how much one can reduce the size of content without impacting quality to a noticeable degree.

Table 1.1: Bandwidth usage for different resolution standards in video streaming. Based upon data from [5, 8]

Standard	**Video Resolution**	**Bandwidth Requirement (in Mbps)**
480p	854×480	2 to 2.5
720p HD	1280×720	5 to 7.5
1080p HD	1920×1080	8 to 12
4K UHD	3840×2160	32 to 48

As such, there is a need to look at improvements within distributed software architectures. Two of the most common distributed software architectures are client-server and peer-to-peer (P2P). These architectures have thus far proven capable of meeting the increasing challenges that have faced distributed applications and services. Each architecture has focused on different areas of how to host and deliver a service, with client-server focusing on how to best utilize servers and data centers, while peer-to-peer has focused on how to best utilize user resources. Each has its own advantages,

which decide its usability for different services. However, both architectures also have its disadvantages and concerns that are ingrained within it.

Security and privacy are something that is a rising concern for users on both architectures. For services with a more centralized architecture, there is a worry about what kinds of data the service collects about its users. The problem lies in how these services use or share their data, since it has a possibility to be used with malicious intent by governments[9] or external organizations[10]. As for the services with a more distributed architecture, there is the potential of users accessing data they do not have permission to access, but are acting as distributors for. Another issue is that users can also intentionally or unintentionally share a modified version of the data, which can be malicious or corrupted[11].

The challenges for today and the future are complex, with distributed architectures such as client-server and peer-to-peer being optimized to face these challenges. However, there is still a limit to how good these architectures can become, and which problems they are able to solve in a satisfactory way. A solution that has been infrequently utilized, or even discussed, is the combination of both architectures into a hybrid architecture. The main idea behind combining these two architectures into one is to solve the challenges of both, while avoiding the main negative aspects that are associated with each of them. This solution could hopefully not just meet the challenges for today and the future, but even improve these services to a new degree.

1.2 Goals and Research Questions

The goal of this thesis is to determine the viability of hybrid architectures for media streaming services. With the challenges media streaming services face moving forward, there exists a valid reasoning to why hybrid architecture shall be looked at and considered. With that said, a proper way to proceed is to focus on hybrid architectures. This thesis will take into consideration how hybrid architecture is utilized in media streaming services, with regards to both benefits and issues. By looking at beneficial and non-beneficial aspects, it is possible to discover significant properties that could be used to solve the existing and coming challenges. To properly perform this analysis of hybrid architecture, this thesis aims to answer the following questions.

RQ1: What properties can be found in services using a hybrid of peer-to-peer and client-server architecture?

RQ2: Which of these properties are significant for media streaming services?

RQ3: What is the viability of a media streaming service using a hybrid architecture?

The first research question entails analyzing services with a hybrid of peer-to-peer and client-server architectures, then identifying and determining properties which are found in the selected services.

To answer the second research question, the properties found in the first research question will be evaluated in relation to media streaming. This requires a thorough examination of the properties, and determining how significant they are for media streaming services and their users. The significance of properties are determined by how important they are for a media streaming service, compared to other services that use the same architecture. The answers will be beneficial in order to understand media streaming services and their requirements, as well as the advantages and disadvantages a hybrid architecture can have for media streaming services.

Finally, in order to answer the third research question, the results from the previous research questions will be evaluated. Through this, the benefits and issues of hybrid architectures in media streaming will be highlighted. This will, in turn, give an indication of the viability of a hybrid architecture in media streaming services.

1.3 Research Method

This research is a case study into hybrid architecture and the impact it may have in regards to media streaming services. The aim is to gain a better understanding of the viability of a hybrid architecture. In order to accurately assess the viability, there need to be a detailed description of advantages and disadvantages with hybrid architecture. This prompted the research strategy to become a descriptive case study, with the environment being hybrid architecture and the instance being media streaming services.

In order to fulfill this case study, there needs to be a varied selection of services that can be analyzed. Considering the focus on specific properties that represents hybrid architecture services, there is a need to select services that have had a certain degree of success with this architecture. It is also useful to find different services that utilize hy-

brid architecture, which can be compared in order to properly determine the difference between aspects.

The selected services will then be analyzed by two different methods. The first method is to go through documents that contains information about the service and how it works. The goal is to find which properties that is significant for each service. The second method is to observe user or third-party feedback, in order to find other potential properties or validate existing ones. This is done mainly by finding relevant documents where users of services have shared their feelings about it.

This means that the thesis needs both qualitative and quantitative results. Qualitative results are needed from the service documentation and the properties that are introduced there. This means that the prevalence of properties is dependent on the source material. However, when finding qualities about the service, it is important to find as many different viewpoints as possible. Therefore, a quantitative result from the user feedback may help give a more nuanced understanding of the service.

1.4 Thesis Structure

Now that the motivation, goals and research method for this thesis have been presented, the remainder of the thesis will be split up as follows. Chapter 2 will go through useful and necessary background information, as well as explaining the approach to the research this thesis is taking. The introduction of the services, as well as the properties that were found, are done in Chapter 3. These properties will be further discussed in Chapter 4. Then, Chapter 5 will evaluate the discussion, in order to answer the questions posed in Section 1.2. After this, the thesis will be concluded in Chapter 6.

Chapter 2

Background Theory & Research Approach

This thesis is written on the basis of how well a hybrid architecture functions for streaming media content, and which properties enables it to do so. This means that there has been a process of selection through determining which aspects should be the focus and which aspects are not necessary. Some examples of these selections are the architectures that have been relevant to explore further, the technologies that have been used in relation to the architecture, which services that have been investigated, and how documents have been selected and analyzed.

Although these aspects are very different, they all fit under the notion that they are needed not only to provide a satisfactory answer to the questions made in Section 1.2, but also to provide boundaries to this thesis.

Relevant architectures, related technologies and important terminologies are presented in Section 2.1. These explanations and definitions are helpful in understanding the fundamental theory behind the thesis, as well as specifying the meaning that these explanations and definitions represents in relation to this thesis.

In Section 2.2, related works to this thesis are presented. These related works are comprised of other academic literature that partly covers the topics explored in this thesis, but does not completely overlap with this thesis.

Section 2.3 covers the selection of services that will be covered in this thesis. It will also split services into categories for definition purposes, and explain why they were selected.

Finally, Section 2.4 will cover how to determine what proper documents are in respect to literature review. This will also explain the difference in scientific articles for qualitative data and user feedback for quantitative feedback.

2.1 Theory & Definitions

Since the topic is an in-depth analysis of a small part of the distributed architecture and streaming field, there might be complications with understanding specific architectures, technologies and terminologies. This chapter is meant to create a fundamental understanding about how these technologies are defined for this specific thesis.

2.1.1 Client-Server Architecture

The client-server architecture is a fairly common and popular distributed architecture. The architecture is built upon a smaller number of servers, and usually a larger amount of clients. Servers provide resources or services to one or more clients, while the clients do not share any of their own resources with the servers or each other. The relationship between them can be visualized as clients send their requests, while the servers respond to the requests. A reason behind its popularity is due to it being one of the most straightforward distributed architectures to implement[12]. A simplified example of a client-server architecture is visualized in Figure 2.1.

2.1.2 Peer-to-Peer Architecture

A peer-to-peer architecture is a set of equal peers that are connected to each other in a way that does not need a central coordination by a server[13]. Peers in a P2P network are both suppliers and consumers, in the way that they are sending what they have and others want, and receiving what others have and they want[14]. A simplified example of a peer-to-peer architecture is visualized in Figure 2.2.

The common ways that peers in a P2P network organize themselves are unstructured and structured. In unstructured networks peers form random connections with each other, and any request for a specific piece of data is propagated throughout the network until peers with it can be found[15, 16]. While in a structured network, peers organize themselves and store information about where a specific piece of data can be found, and can share this information directly with the requesting peer, even if they do not have this data themselves[14, 17].

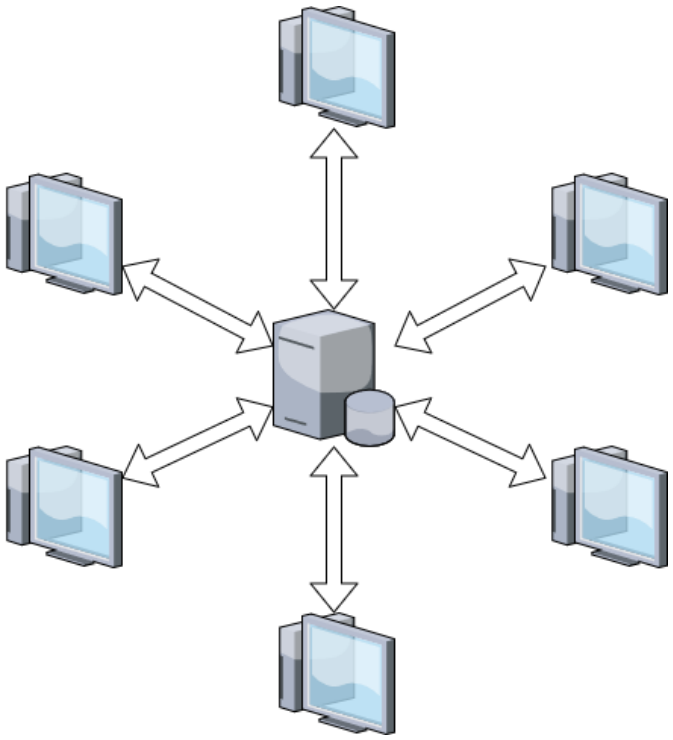


Figure 2.1: A simplified example of a client-server architecture.

2.1.3 Hybrid Architecture

A hybrid architecture is an architecture that is a combination of other architectures. There are multiple ways to define a hybrid architecture, as there can be several possible combinations of existing architectures. However, due to the thesis focusing on distributed services and software architecture, the hybrid architecture definition is limited by these conditions. For this thesis, the focus on hybrid is going to be on the combination of a P2P architecture and a client-server architecture. Within this type of hybrid architecture, there could be major differences in how they are implemented or how they work.

So for this thesis, the definition of hybrid architecture will be any combination of P2P architecture and client-server architecture. This is because there is not a particular need to further divide the term hybrid architecture into smaller definitions, even though it is possible.

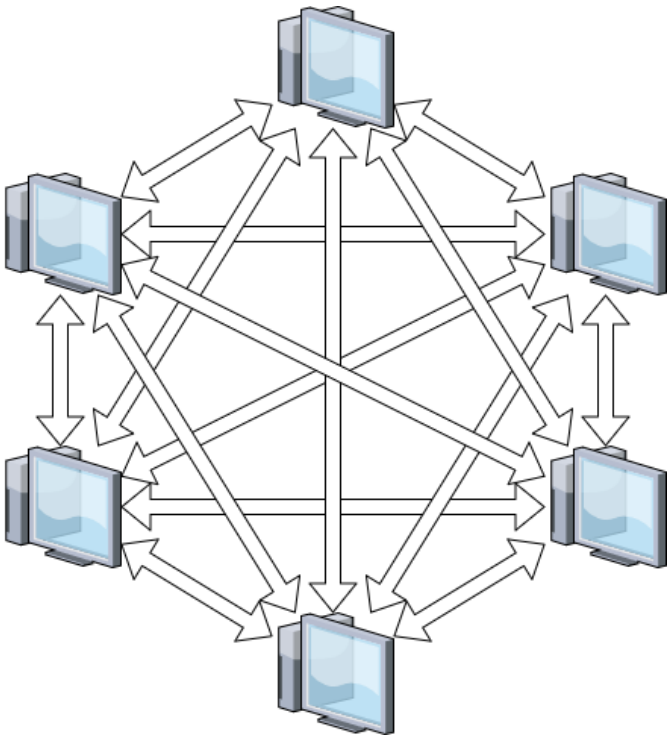


Figure 2.2: A simplified example of a peer-to-peer architecture.

2.1.4 Content Delivery Network

Content delivery networks (CDNs) are a distributed network of servers, which are used to serve most of the content on the Internet today. These solutions provide high levels of availability and performance to services, and thereby clients, by having points of presence (PoP) around the world. The clients are usually connected to the best available PoP for them, based on a selection criteria that usually consists of any combination of lowest ping, highest performance or lowest cost for the CDN. These PoPs usually act as caching servers, by storing duplicates of the original content to be served to clients in their respective geographical regions[18].

An early example of a CDN is Akamai, which was publicly released as early as 1999. Their business model consisted of owning and maintaining servers all over the world, which would mirror content from their customer. This would enable everyone who wanted to look at a website, to be connected to a closer server, rather than spending more time getting the same information from a server on the other side of the world. Not only did this help services with last-mile delivery, but it could also help with maintaining availability since it could reroute clients to another server in case of attacks[19, 20, 21].

2.1.5 Cloud Computing

Cloud computing is a relatively modern aspect of computer services, and is based around providing storage and computational power to services on demand. It works by having server farms located at various points around the world that are usable for the organizations that subscribe to the cloud service. Although there are clouds that are made specifically for one single organization, this thesis will focus on public clouds that are used by multiple organizations.

A cloud platform typically has prebuilt services that may or may not be of use to an organization. When an organization require a service from a cloud platform, they would have to subscribe to that platform's services. Although organizations subscribe to the cloud, many clouds are working with a "pay as you go" model, enabling organizations to only pay for what they need and use[22]. Multiple different organizations commonly host their services in the cloud, sharing the server resources available there. This is achieved through virtualization technology, where computing resources are abstracted from the physical servers and combined into virtual nodes, which only reserves the resources it needs based on the task running on it. A software arbiter controls how resources are to be divided between virtual nodes, and in turn the organizations hosted there[23].

2.1.6 Streaming Media

Streaming media is used to describe content that is being presented to the user while it is still being transmitted by a provider. It is different from file downloading, where the user has to download the entire file before it can be presented to the user, which was common in the start of the 2000s and earlier, thanks to services like BitTorrent[24]. Although these services were popular and helped define the P2P architecture, the focal point of this thesis is on streaming media, thereby rendering these services outside the scope of the thesis.

Today, streaming media is one of the most common ways to transmit media over the Internet. A few examples of streaming media are Spotify, YouTube and Netflix. Streaming media can be divided into two categories, which are on demand and live. On demand is when the entire content is already available to the user, so the user can play, pause, rewind or fast forward to any part of the content at will. Examples of this is a movie or a song. Live is when the content is being delivered to users as soon as it is produced, so the user is not able to fast forward. With examples being a live stream of a concert or a sports event[24].

2.1.7 Properties

For this thesis, properties are defined as follows. Properties are quality attributes that are characteristic of a service or architecture. The selection of quality attributes was decided by determining which attributes were not just noticeable, but also somewhat defined in the service or architecture. As an extension to the definition of properties, the thesis also recognizes attributes that are not commonly regarded as quality attributes, but are considered closely associated and linked to them. Furthermore, while these attributes could focus on positive qualities, they could also point out negative ones. Negative attributes are equally important, as they also are noticeable and define a service or architecture.

2.2 Related Works

An article with partial coverage compared to this thesis is covering cost-effectiveness in distribution of streaming media content. This article takes into consideration a combination of a CDN and a P2P network, and issues with cost of CDN, peer heterogeneity and resources available from peers. Although the data is simulated and not put into practical use, the verdict is that the hybrid architecture could be both cost-effective and scalable[25]. The main differences between this thesis and the article is that the system was developed and tested in a experimental setting. As such, the results might not be completely accurate in relation to a practical setting. Another difference is that this thesis focuses on multiple services, while the article focused on a singular system.

Another article that were considered partially coverage focuses on P2P-VoD streaming. Although the focus of this article is more closely resembling streaming media content via P2P, there is also included aspects about a combination with CDN. Nonetheless, this article mentions the challenges posed by using a P2P architecture for video on demand (VoD) and the issues that come with it, such as asynchronous arrival of peers, peer churn and lack of scalability. The general verdict was that it is possible to maintain a singular P2P architecture to keep costs down without losing scalability, but many sources within it lean towards a hybrid architecture in order to maintain best possible scalability[26]. This article differentiates from the thesis in regards to the focus. The thesis focus on what properties make hybrid architecture viable, with the article focusing on challenges facing a pure P2P-VoD system.

2.3 Service Selection

The services that should be the focal point of this thesis are selected based on what they cover. It is very important that the services are representative in a broad sense to streaming media or a hybrid architecture. Furthermore, the services need to have enough documentation, information or feedback from users to warrant further investigation.

Another aspect is to select services that have developed over time, in order to judge not only their approach to hybrid architecture, but also judge the amount of success they have had. Based on these factors, it would then be possible to find certain properties that are linked towards using hybrid architecture for streaming media. Then, these properties could be put into context with popular streaming services, to determine how hybrid architecture could impact those services.

With these aspects in mind, the selection of services have been divided into four categories where each part plays their own role in this thesis. The four categories that are considered to be important are based on technological advancements and how the services are utilized. They are separated by services approach to hybrid architecture, what kind of challenges the service faced, and what the service delivered with the help of hybrid architecture. The services introduced in Sections 2.3.1, 2.3.2 and 2.3.3 will be further looked at in Section 3.1. All categories and their associated services are shown in Table 2.1 to provide a brief overview.

Table 2.1: Service categories and their associated services.

Category	Services
Distributing Services	Skype Pando
Hybrid Architecture Streaming Services	Spotify Xunlei Kankan
Live Streaming with Hybrid Architecture	SLIVER.tv
Mainstream Media Streaming Services	Netflix YouTube

2.3.1 Distributing Services

The first category is distributing services. These services are examples of distributors using hybrid architecture, although they are not focused specifically on streaming media. What is separating this category from other categories is the way that these services have implemented their hybrid architecture, as well as being services that are not strictly related to streaming media.

For this category, the services selected are Skype and Pando. Both of these services were developed for a specific purpose, and have been using hybrid architecture, in order to fulfill that purpose.

2.3.2 Hybrid Architecture Streaming Services

Some cases of streaming media content with hybrid architecture are the focal point of category number two. These services use the same principles as the services in the first category, but utilize these principles to exclusively stream media content.

Therefore, the selected services landed on Spotify and Xunlei Kankan. These services represents two different ways of utilizing hybrid architecture to stream media, at approximately the same period in time.

2.3.3 Live Streaming with Hybrid Architecture

This third category represents a modern take on hybrid architecture for a streaming media service. The selected service has used the experience and knowledge garnered from earlier, similar services in order to improve their own service. This service also provide live streaming, instead of on demand streaming that the earlier services provided.

Due to this definition, the selected service is SLIVER.tv. This is a service that has worked out how to enable users to become peers in a hybrid architecture to deliver media content.

2.3.4 Mainstream Media Streaming Services

In order to determine the viability of hybrid architecture for streaming media content, the benefits need to be compared to existing streaming media services that are not using a hybrid architecture. These services make up the fourth and final category, the services that are selected to represent the existing non-hybrid architecture services.

The selection of services in this category came from how popular the services are, and how large these services are. Therefore, the selection is Netflix and YouTube. The benefit of selecting some of the largest providers of streaming media content is that there are already existing information and research on it. However, as these services have never used a hybrid architecture, or have any official plans to do so, they will only be used for comparisons or as services that could potentially transition to it. Therefore, they will only be presented briefly here.

Netflix is one of the largest video on demand streaming services on the market. It is hosted on Amazon Web Services after migrating from physical data centers[27]. A large reason for the migration being that it was more cost-effective than trying to maintain their own servers. Another reason was that due to popularity, Netflix could not keep up with the capacity increase and needed a solution that was more scalable[28].

Similarly, YouTube is also one of the largest streaming platforms on the market, providing both live and on demand media content. With a billion hours watched every day, the platform is responsible for 37% of mobile data usage[29]. Due to the sheer amount of videos available as well as all the space they occupy, the need to have an efficient way of distributing it is important.

2.4 Selection of Documents

Every service will be explored in the sense that they will have an explanation of the architecture, and how it is used. Most of this explanation will be from technical documents about the service. However, some services are closed source, and explanations might rely on third-party articles and observations to make an educated guess, as to how their architecture was built and works. This is especially true when looking at Skype, Pando and Xunlei Kankan. This means that there might be errors in the explanations of the actual architectures in this thesis, which also leads to errors in the discussion later. However, this and other issues will be addressed further in Section 5.4.

When it comes to the users experience with the service, the vast majority of these are gathered by finding appropriate platforms, and reading through their feedback. Although this gives a unique view of the service, it also means that many of the documents that are gathered about users are fairly subjective. However, the scientific significance in this regard is the quantitative data that can be gathered from this user feedback. It is also important to make a note of the different feedback gathered in this fashion, as some of the documents come from official questions answered by employees of the service, while other documents are from personal blogs or forums.

Chapter 3

Services & Properties

This chapter is going to present the services that have been introduced in Section 2.3. It will describe how these services developed their architecture, as well as experiences they have had due to their choice of said architecture. Other relevant and interesting aspects that have been discovered will also be elaborated here. This presentation of all the services will lead to a selection of properties that have been found to be relevant to these types of services. These properties will then be introduced and explained.

Due to the progression of the chapter, it will be divided into two sections. Section 3.1 will focus on the services and their presentation. The properties that are found here will be presented in Section 3.2.

3.1 Services

This section describes how services have utilized a hybrid architecture, and what challenges they faced or benefits they reaped from this choice. It also covers other interesting aspects about the services that are considered relevant for this thesis.

3.1.1 Skype

Skype is a P2P-based messaging service that first was released in 2003. The service was originally intended as a voice over IP (VoIP) service, but evolved to also include messaging and file sharing functionalities. The service became known as an important

tool in the communication sector, and was used by businesses and private households alike.

Even though the system was considered to be P2P, and that they had mainly P2P functionality, two reasons point towards it being a hybrid architecture. First, it had a central server that contained the user names and passwords of users, which was used to identify nodes. Secondly, some nodes in the network could become super nodes, which functioned similarly to servers. Every node in the Skype network were measured by whether they had a public IP address, and sufficient CPU, memory and bandwidth. The nodes that fulfilled those criteria could be selected to become super nodes. Super nodes functioned as relay points for other nodes, thereby also knowing their location[30]. There were also arguments about the requirements to qualify as a super node, mentioning that bandwidth and public IP address were enough[31]. These two reasons were used to determine that Skype used hybrid architecture. Since the service was closed source when originally released, the design of the architecture is a proposed design from an article about the service, which is displayed in Figure 3.1.

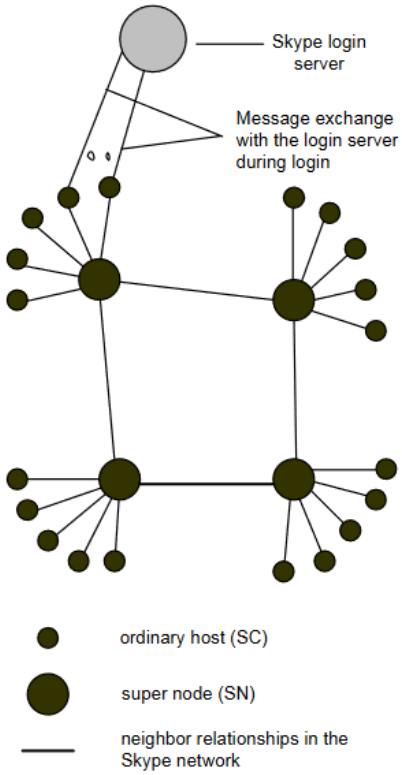


Figure 3.1: Proposed design of Skype P2P network. Displays the three types of nodes in the network[30].

The service was not perfect, and a large issue entailed super nodes and how they were selected. Due to the seemingly random selection of acceptable nodes, many of the selected nodes were located in universities, with one source claiming that up to 35% of nodes were universities[30]. This was not popular though, with universities banning the use of Skype on campus[32]. Some of the reasons was that Skype could easily be used by malicious users as a backdoor to corporate networks[32, 33], Skype was virtually untraceable[33], and also that Skype requiring users to grant it "usage rights" on the network, which users did not have[32]. Another important aspect was the bandwidth, with a standard node utilizing very little data, but super nodes using a significant portion of the bandwidth, considering the time the post came out[32].

Thanks to issues like this, and improvements in technology, there was an interest to transition away from hybrid architecture. According to the principal architect of Skype, they started to transition their product away from P2P in 2009[34]. This transition continued even after being acquired by Microsoft[34, 35]. The main reason this transition happened was due to a software bug that took out a large portion of the P2P network, and leaving the remaining nodes to collapse[34, 36]. Due to these bugs happening to super nodes that were hosted by the users, it became difficult for Skype developers to remotely maintain them, prompting them to switch to dedicated servers. Another big reason was due to the large increase in mobile units, that could also work as peers. Not only did they have to constantly work as nodes, using processor power and draining battery life, but they would also be using large amounts of cellular data[34, 37].

Shortly after, Microsoft announced that they would further transition Skype into a cloud service using Azure, and at the same time removing the final P2P pieces from the service[34, 38, 39]. The reasoning for this change was to support mobile environment better, as it was a fast growing user base[34, 38]. Another reason was that with cloud servers, messages would be put on hold and delivered when the recipient was online. This was opposed to the previous P2P architecture which needed the recipient to be online in order to get the message[36]. The decision to move to a cloud service was to improve the user experience[34].

Although Skype has been a large part of telecommunications since it released in 2003, there have been cases where Skype was accused of bad behaviour. Not just by the universities blocking the service, as mentioned earlier, but also the removal of trust when Skype shared call information with the American government[37, 40]. After the news went public, several outlets considered this transition suspicious[36, 37], and some users commenting that this case broke their trust in Skype[36].

3.1.2 Pando

Pando Networks was one of the earliest companies attempting to create a hybrid of client-server and P2P. In 2008 they deployed their commercial service with success, gaining NBC as a customer for video streaming[41]. The following years they also gained multiple customers from the gaming industry[42], such as Riot Games, the creators of popular game League of Legends[43]. Pando provided the technology allowing customers to connect Pandos service with their own CDNs. Which meant that Pando would handle the client coordination and P2P aspect for the customer. The core principle of the service was to dynamically switch users between using CDNs and P2P for downloading, leveraging users to share what they had already downloaded between each other. What this meant in practical terms were faster download speeds for the users, reduced expenses surrounding hosting for the customer, and less complexity for the customer[41, 44].

While it would seem like Pando was an inherently positive thing, there were also many users who complained about it as well. For example, some users in Australia had their internet speed throttled or got large bills due to Pando running in the background. This problem arose as Australian internet service providers (ISPs) had limits on how much you could upload each month. Others complained about erratic experience with the service whenever Pando was active. An important issue was the lack of transparency by the companies utilizing Pando. Usually Pando was installed in the background during installation of the service software, without any notifications or request from the user to accept said third-party installation. Not to mention that there were few, if any, references to Pando in the terms of service or similar documents during the setup of the service software. Overall this lead to Pando receiving a somewhat negative reputation from users[45].

However, the software in itself was, by some, considered trustworthy[46]. Then again, the lack of transparency made some users vary of using the software[47]. These situations impacted the credibility of Pando, and may have reduced the overall usage of the service. Additionally, security concerns were raised about Pando, as antivirus software noted Pando as potentially suspicious, or even malware, due to how Pando worked. The service became flagged as malware to such a degree, that firewalls started to block the application[48, 49, 50], which became an issue since the service needed to be allowed through the firewall to function[51]. Another security risk was that the service needed permission to monitor other applications, which did not help their credibility[52].

Although there were multiple references to the negative aspects of Pando, there were also some positive developments that came from Pando. In 2007, the P4P Working Group (P4PWG) was created by Pando Networks and Verizon Communications, in order to improve P2P effectivity and quality[53]. P4P allowed ISPs to integrate an

iTracker into their network architecture. The iTracker would be hosted by the ISP, in order to provide information about network topology, as well as location information about peers. This information could then be used by P2P applications to make educated choices in peers to connect to. Which would prefer peers within the intranet of the ISP, or peers that were geographically close. This would benefit users of P2P, as well as ISPs as they could reduce the costs related to poor network routing that plagued P2P[54]. A visualization of P4P compared to other network architectures is illustrated in Figure 3.2.

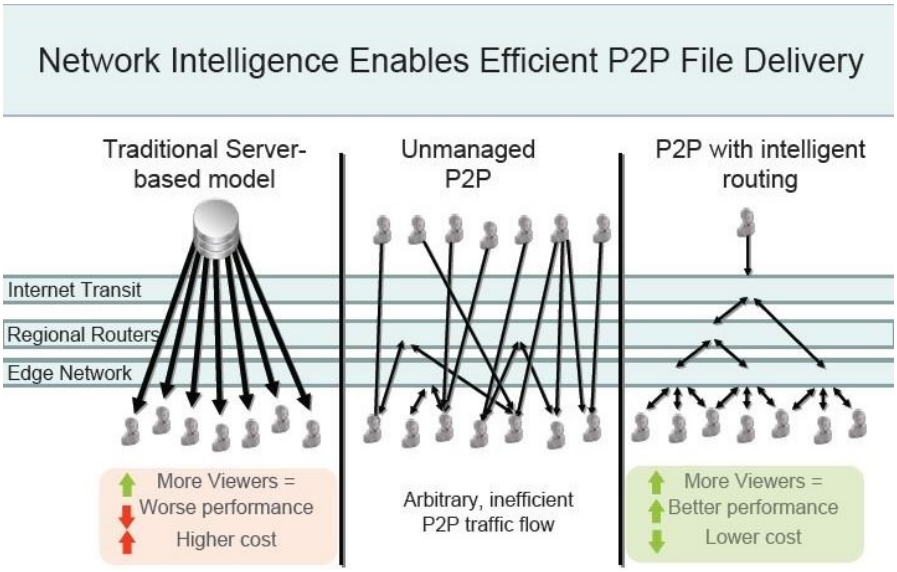


Figure 3.2: A comparison of network architectures with pros and cons. P4P can be seen on the right side. Adapted from [55].

In 2013, Pando Networks was acquired by Microsoft Corporation[56], with speculation that Pando technology was going to be used in development of the new Xbox[57]. Later in 2015, Microsoft announced it planned to deliver most of Windows 10 updates through P2P, which The Verge surmised was a product of the acquisition of Pando[58].

3.1.3 Spotify

A streaming service that started with a hybrid architecture was Spotify[59, 60]. Spotify is a service that provides music streaming on a global level, although they had more moderate beginnings. Spotify utilized a peer based sharing system, in order for users to share music files with each other for the purpose of achieving non-buffered and seamless streaming[59, 61, 62]. P2P technology also had the added benefit in reducing costs by

removing the need to maintain a larger quantity of servers, making it ideal for a startup company like Spotify[59, 61].

When Spotify launched in 2008, they utilized servers as storage for music tracks. The distribution of the tracks were to a large degree handled by a P2P network. Spotify separated their servers into two different levels. The first level was the main storage that kept all tracks in their database, and the second level was distributing servers which kept a smaller number of tracks, but distributed the ones they had into the P2P network when requested. The P2P network was used to share the music tracks with each other to alleviate the second level of servers[60, 63]. This architecture is visualized in Figure 3.3.

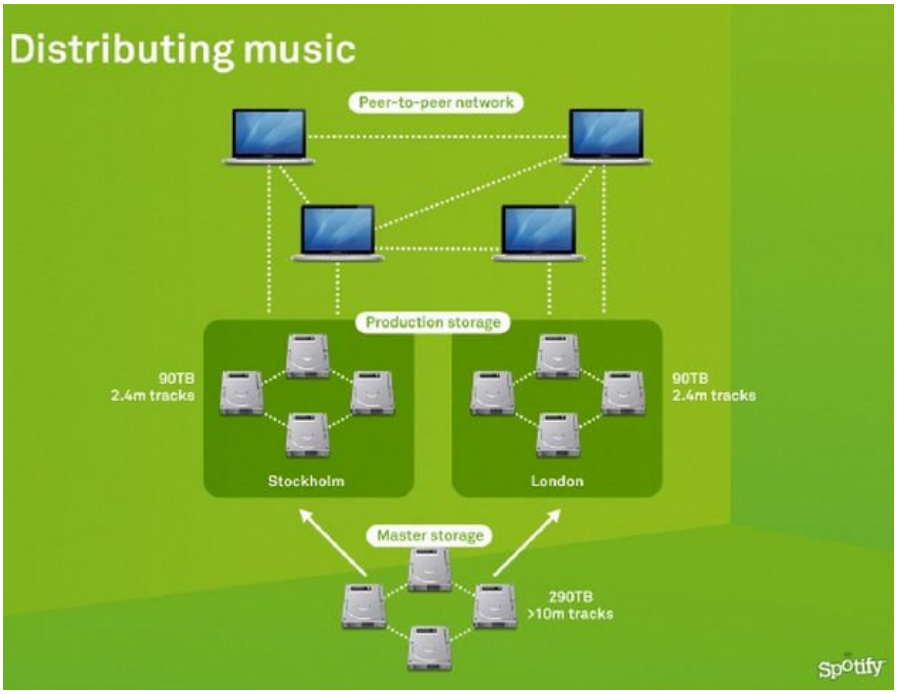


Figure 3.3: Theoretical design of Spotify's hybrid architecture. Displays the three levels of the architecture[63, 64].

As seen in Figure 3.3, the two levels of the servers were named "Master storage" and "Production storage". As previously mentioned, the "Master storage" was the primary storage of music tracks. It worked with a distributed hash table that could look up the correct value, in this case a musical track, based on a hashed key. However, this server only found and distributed these tracks if an end user could not find the track in the "Production storage". The "Production storage" was a faster, distributed server that worked as a cache with enough RAM to quickly find and distribute tracks to the P2P

network. Due to these tracks being distributed, the users would find that there was less latency when accessing them[63].

Then there was the P2P network, which was made with a custom protocol[63]. The client requested a song and began by checking the cache of the client's unit, and played it from there if it existed. If not, the client asked the P2P network to find the song and transfer it to the client. While waiting for the chunks each peer transferred, the client requested up to 15 seconds of the music track from the server, so it would not experience loss in content while waiting for the rest of the track. The client used all sources in order to enable seamless streaming. However, if the client knew the next track, it would search for the song among the peers approximately 30 seconds before the end of the current track. This is so the next track could be played without issue. If the client selected a new track from another place, it would simply repeat the first process over again, thereby always being able to provide non-buffered streaming[60].

In the beginning of Spotify, they had periods where they had to limit registration of new users for the free tier. This was due to them wanting to control the growth of the service, and avoid any unexpected problems that could arise from this. These problems could, for example, be performance or scalability issues that could appear due to a massive growth past their capabilities[65, 66].

During 2014, news outlets reported that Spotify were switching their architecture away from P2P architecture[59, 61, 64, 67]. Spotify felt that their growing number of servers were a good reason to stop utilizing a hybrid architecture, because they could still supply a seamless experience without needing P2P[59, 61, 64]. One outlet even mentioned that with a growing number of servers, maintaining a P2P network would be an unnecessary expense[67]. Spotify also mentioned that switching away from P2P technology made it easier to enable a "family" mode, where different users could listen to different music on the same account simultaneously[59]. Another outlet mentioned that this switch would prevent excessive use of bandwidth[61], although another outlet mentioned that the bandwidth requirement was not as intensive as with video streaming[67].

Spotify announced in 2016 that they would stop running their centralized servers and start using Google Cloud Platform (GCP)[68, 69, 70]. Although this transition happened only two years after moving to server farms, officials from Spotify argued that cloud platforms were not developed enough at the time and the safer choice were in server farms[70]. Other arguments were the cost and time it took to maintain the server farms[68, 70] and the benefit of valuable cooperation with Google employees[69, 70]. This also gave Spotify access to Google's Big Data technology like BigQuery[71], which became a huge benefit to Spotify employees[72, 73].

Although Spotify has benefited greatly by using P2P technology, it has also led to some problems. Some users experienced increased bandwidth usage[74, 75] or throttling from

their ISP[76, 77]. These situations could have been avoided if Spotify adequately disclosed their use of P2P distribution[62, 74]. These problems have since been addressed by Spotify, but the impact of these issues on the reputation of Spotify is uncertain.

3.1.4 Xunlei Kankan

Xunlei Limited, a Chinese corporation, started out by providing P2P assisted acceleration of file transfers. Xunlei has since expanded with a large variety of services for its users, but a commonality between them is that they all utilize a combination of peer assisted and server assisted downloading, essentially a hybrid architecture[78]. However, their most interesting and popular service was their VoD service Xunlei Kankan[79], which launched in 2007[80].

The front end of Xunlei Kankan consisted of a client application that worked as both a streaming application, as well as a peering application[81]. When a user watched something in the client application, the content was also stored and cached, in order to allow further P2P distribution. The content would later be deleted when the user closed the application[78]. The back end consisted of three parts, which were the Kankan portal, that contained information about the videos and the location of the content in the system, the Kankan CDN, which contained the actual video files, and finally, the Kankan P2P network, which contained the trackers for the P2P service[81]. The overall design of Xunlei Kankan can be seen in Figure 3.4.

The way the service worked was that the user would look for a video on the portal. Then, when the user decided on a video, the client application would query the CDNs as well as the P2P network. To ensure quick initialization for the stream, Kankan utilized a secondary CDN that only contained the first segments of a video, which it would start streaming from. Then when the primary CDN and P2P had been configured on the client, it would start streaming from either the P2P network, the CDN or a combination of both, depending on the availability of the video. The selection of streaming source is chosen by the client application based upon quality of P2P and whether or not enough of the future parts of the video are cached[81].

In 2013, Xunlei employees, who have since been fired, made a program which would install malware on Android phones connected to the computer. This issue was not Xunlei Kankan's fault, but since they use the same infrastructure, it highlighted a security flaw in their architecture. Another point that caused problems was that the program was named Kankan. This meant that even though Xunlei Kankan was not involved, their credibility was affected simply because the program used their name[82, 83].

A study focused on Kankan found that it only utilized a few hundred CDN servers, compared to the thousands that similar services such as YouTube used[81]. The same study

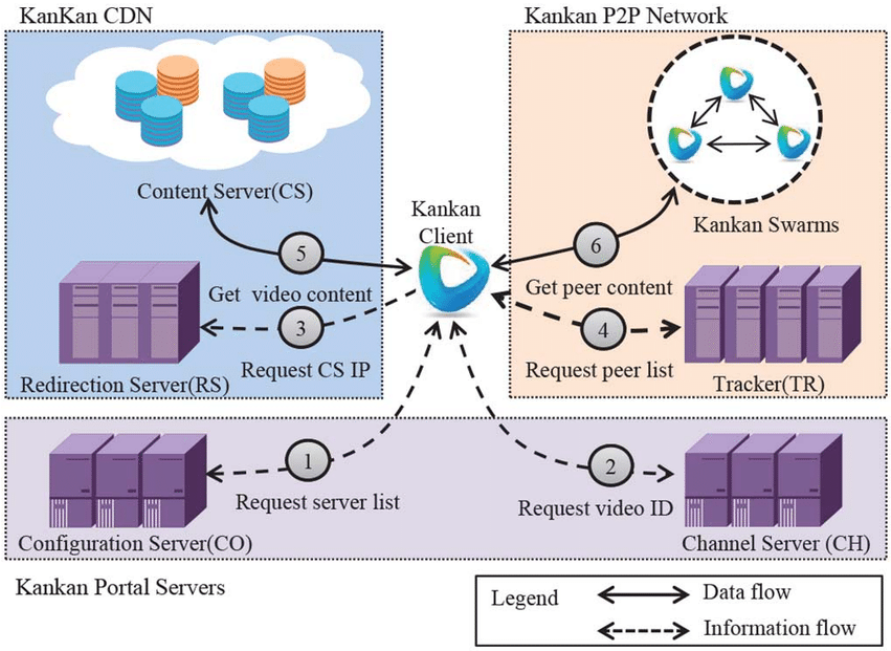


Figure 3.4: Illustration of Xunlei Kankans architecture. Adapted from [81].

also found that for popular videos, 98% of the content was delivered through the peers. In 2015, Xunlei sold their Kankan stake to Beijing Nesound International Media Corp. Ltd. due to wanting to focus on some of the other services they provided[84].

3.1.5 SLIVER.tv

SLIVER.tv was founded in September 2015[85], providing a platform for live streaming video games in virtual reality (VR), in up to 360° format, for both streamers and viewers[86]. At the start of 2018, SLIVER.tv officially announced Theta, a blockchain based around video streaming[87]. Theta Network, formerly known as Theta Token, wanted to change how video was delivered over the Internet, by utilizing viewers and volunteers for content delivery and transcoding of video. THETA would be used as a cryptocurrency reward for those sharing their network or computer resources. The goal behind Theta Network was to solve some of the problems around current video streaming, as well as improve on preexisting solutions[8].

The decentralized architecture of the Theta Network is quite similar to the traditional ways of handling live streaming. The main difference is that viewers and volunteers provide their own resources, to act as a CDN or ingest server. Theta Network completes this by segmenting the users into different node categories. Influencer nodes (streamers)

stream their content to the ingestion nodes, which will transcode the content to the different formats. Then the caching nodes fetch the transcoded content and propagates it further to either other caching nodes, or directly to the viewer nodes[88]. The overall proposed architecture can be seen in Figure 3.5.

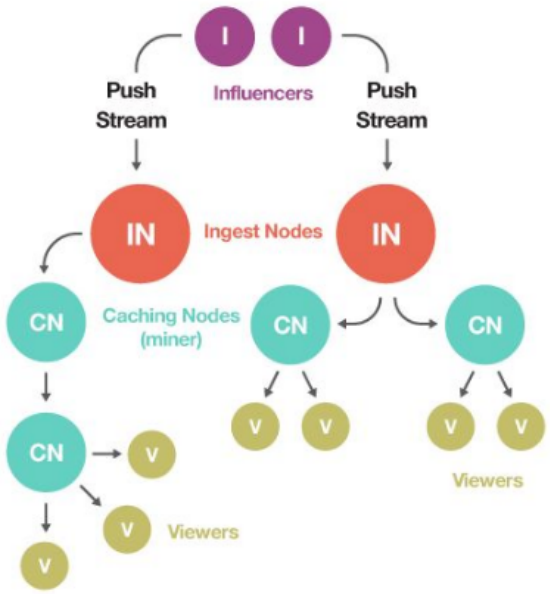


Figure 3.5: The proposed Theta architecture. Adapted from [88].

Users can become either ingest or caching nodes by installing specialized software that enables this functionality. A user can be a viewer simultaneously while being a caching node. Due to the live stream nature of the video, video packets will usually be sent in order, where the urgent high priority packets will be sent first. Furthermore, a caching node will also cache the entire video stream, in order to be able to serve all viewers and reduce rebuffering of the video stream. When a viewer or caching node enters the Theta Network, it will start to locate peering nodes that are physically nearby. The peering system uses a distributed hash table, where each node has a unique global user identifier (GUID). The GUID consists of a 64 bit string, which identifies the region the peer is located, as well as a 32 bit string, which is completely random, in order to differentiate between peers within the same region. Once a candidate list of peers has been generated, it will ping each node to measure the actual latency. A node will prefer peers with lower latencies, as it implies the peer is physically closer and of better quality[88].

Theta Network has a goal to solve the problems with last-mile delivery, due to its highly distributed nodes. Furthermore, it hopes to reduce the large hosting costs related to video streaming services, by utilizing the redundant resources of users. These reduced

costs will in turn leave more revenue for the influencers, as well as the users. Due to the blockchain nature, it also allows advertisers to follow the transactions, and see which influencers played their ads, or which users have been watching their ads. Additionally, SLIVER.tv's web player utilizes Web Real-Time Communication (WebRTC), which essentially provides a simple application programming interface (API) through JavaScript, that enables audio and video to be delivered through peer-to-peer in web pages. As it is natively supported by most browsers, this means users do not have to install any third-party addons to the browser. Which is a great development for services that desire to use P2P for media streaming[8, 89].

While SLIVER.tv desires to have a fully distributed architecture in the future, it is currently utilizing a hybrid architecture. As it needs to build up a user base that can support the fully distributed Theta Network architecture. The main difference is that it utilizes centralized CDNs and ingest servers, with PoP, to ensure availability and performance of the system, with peers offloading the centralized servers[88]. The current SLIVER.tv architecture can be seen in Figure 3.6.

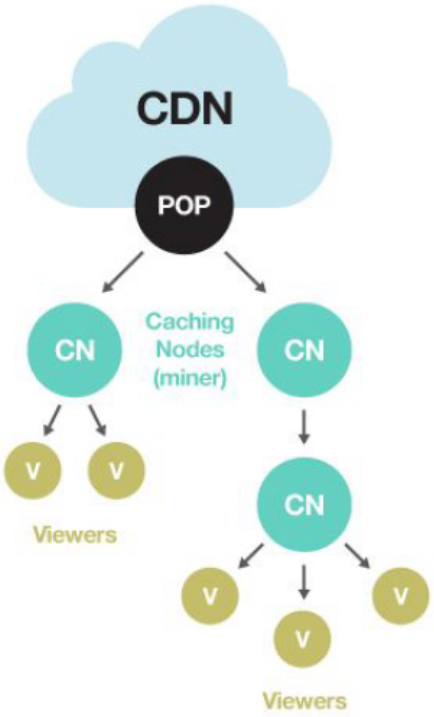


Figure 3.6: The actual SLIVER.tv architecture. Adapted from [88].

A problem that came to light about SLIVER.tv, was some security issues they had with bots and fraudulent behaviour. These bots and fraudulent users managed to abuse the virtual currency system, gaining benefits over legitimate users of the service. In an

attempt to solve these security issues, several users had been flagged and banned by the system. Some of these users claimed they were legitimate and had been banned by a mistake, and as such, were rather unhappy with this development[90]. Another problem was the legitimacy of SLIVER.tv, as some users were concerned with how real money could be exchanged for virtual currency, but virtual currency could not be exchanged back for real money[91]. At the same time, other users were concerned with how the rewards you could buy with virtual currency were based on randomness, which was more reminiscent of gambling than actually redeeming a reward[92].

3.2 Properties

This section will cover the properties that have been discovered within the services presented in Section 3.1. The properties will have a short explanation combined with examples of said property from the different services, where applicable.

3.2.1 Cost-Effective

A property that has been clear throughout Section 3.1, is cost-effectiveness. The property can be defined as a service having reduced costs compared to other similar services not utilizing a hybrid architecture, in combination with fulfilling the requirements and meeting constraints that are placed upon such an service. Potentially even achieving better performance and reaping benefits from it. An illustration of the concept can be seen in Figure 3.7.

The early architecture of Skype was very cost-effective, due to most of the architecture being focused on P2P. The super nodes in the Skype network provided great performance, at minor cost for Skype itself. Very little was centralized, and this meant Skype could actually offer the service for free to all users, with some extra functionality being available through payment.

Similarly, Pando also have some points that corroborates cost-effectivity. Most services that utilized Pando confirmed that using it was cheaper than having to host and maintain a larger delivery network on their end. Not to mention that these services almost unanimously agreed that their service had greater performance after the switch to Pando was made.

Spotify also gained cost-effectivity with the architecture they used during their startup phase. Thanks to it, they could deliver seamless streaming at all times, while having less costs thanks to requiring fewer distribution servers. In a similar fashion to Skype,

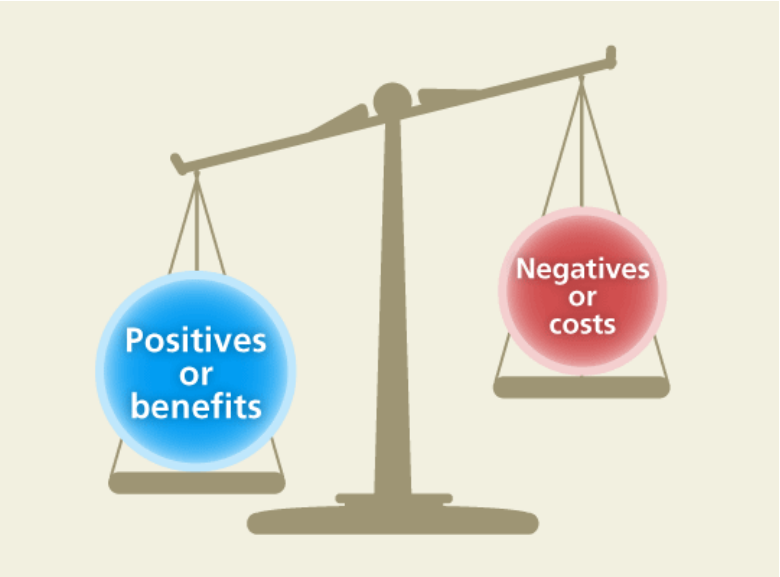


Figure 3.7: Illustration of how benefits ideally should outweigh costs. Adapted from [93].

Spotify could also offer the service for free to its users, albeit with some constraints on functionality and quality.

Taking a look at the section about Xunlei Kankan, there are a few interesting points around cost-effectivity that can be identified. The most interesting one is the claim that they managed to provide smooth streaming of a video, where 98% of the data was transferred over the P2P network. Furthermore Xunlei Kankan apparently only utilized a few hundred CDN servers in comparison to the thousands that other large video streaming services did at the time. Which only further provides evidence around it being cost-effective.

Finally, SLIVER.tv reasoned that using traditional CDN streaming today, was not sufficient to achieve stable, low delay, and cheap enough high-definition 360° VR streaming. As the service is quite new compared to the other services that have been explored, whether or not it is actually cost-effective, is something that will have to be seen for the future. However, considering the influential people who are either backing or working with SLIVER.tv, such as several key people from services such as YouTube, it is unlikely that their shared experience and knowledge does not provide credibility to it being cost-effective.

To summarize, all of the services that have been explored have had examples of cost-effectivity as a property. There are some similarities between how they have achieved cost-effectivity, as well as certain differences between the services as a whole. How

cost-effectivity could potentially impact these services for better or for worse, will be discussed in Section 4.1.

3.2.2 Scalable

Another property that was discovered after analyzing the services in Section 3.1, was scalability. For a service to be scalable, it needs to be able to increase capacity as resources are being added to it. Essentially, allowing more users to use the service concurrently, without severely impacting their experience of the service[94]. An example can be seen in Figure 3.8. Resources in this context can be RAM, CPU and bandwidth. There are two approaches to scalability, which are vertical and horizontal scaling. Vertical scaling consists of adding or removing resources to a single node, while horizontal scaling consists of adding or removing nodes from a system[95]. As such, a hybrid architecture could allow great flexibility in the approach to scalability.

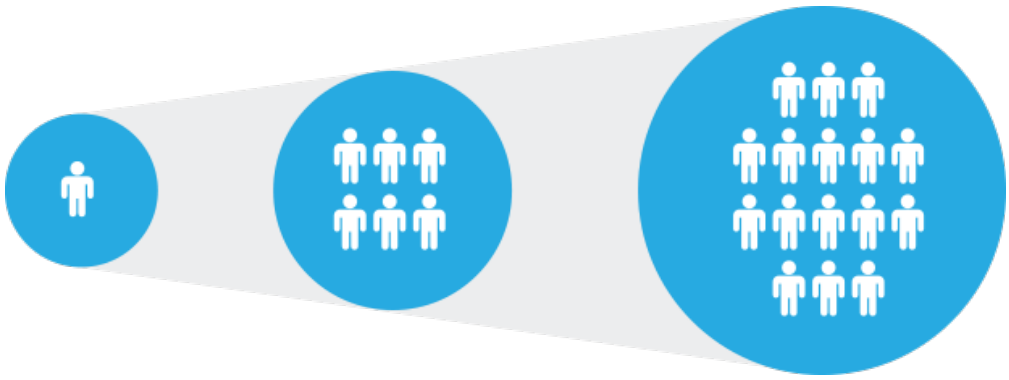


Figure 3.8: Illustration of how scalability should increase capacity of a system. Adapted from [96].

Skype's architecture was based upon horizontal scaling due to their dependence on P2P, as they had a selection criteria for peers to be used as super nodes. As the service grew, new super nodes would emerge to handle the growth. However, this scalability was quite fragile. In one case, problems were caused by the resources Skype consumed from the users, which got it banned at certain locations and institutions. Another case happened when a large portion of the P2P network was knocked down by a bug at one point.

For Pando, their customers had issues with scalability due to the nature of the service the customers provided. When the customer released something new in their service, it was as though a flood gate opened, and their servers were overloaded by the users. Due to user activity at most times being low enough for the customer to handle on their own,

and a lack of funds or desire to upgrade servers, Pando offered the perfect solution for them. Thanks to adding and handling P2P for their solutions, these services had less problems meeting the demands of their users during peak user activity.

Looking at Spotify, they managed to scale their service to a great degree. Their three layers of distribution optimized track retrieval for their users. Thanks to allowing users to download tracks amongst each other through their P2P network, this helped relieve their servers during periods of large activity. This allowed horizontal scaling as the service grew. At the same time, their centralized servers provided scalability when the load was lower or when the user requested an unpopular track. The servers could also be upgraded with more resources which would enable vertical scaling, and new servers could also be added providing horizontal scaling. The methods and techniques Spotify used in their solution, such as always streaming the first 15 seconds from servers, benefitted the scalability of their service.

Xunlei Kankan utilized similar methods and techniques as Spotify, for their solution to achieve scalability. The main difference being that it was provided for a video streaming service. While Spotify had used both production and master storage to handle initial streaming, Xunlei created secondary CDNs that only contained the first thirty seconds of a video. This gave the client ample time to initialize the P2P or primary CDN connection to access the content. The extra layer prevented long initialization times for streaming, as well as enabling scalability. It was a choice that turned out to be good for the service, considering video streaming can be more complex and resource taxing than music streaming.

Since SLIVER.tv are delivering live streaming of video in a 360° format, they felt current cloud technology would be insufficient or too expensive to scale well enough. As such, they created client software that would be able to handle all the tasks that commonly were only handled by servers, such as transcoding and delivery of content. Due to the blockchain revolution, there were a larger degree of machines that would be sufficient to complete these tasks on behalf of SLIVER.tv. In combination with this, as well as the potential lower distance between peers, SLIVER.tv determined the service could achieve greater potential levels of scaling than a cloud platform.

As can be seen, scalability is a property that was identified in all of the selected services. While there exist differences and similarities between them, the common aspect of the services is the hybrid architecture. Scalability as a property of a hybrid architecture will be discussed in detail in 4.2, where potential issues and other interesting points will be accentuated.

3.2.3 Complex

Complexity is a property that became apparent during the analysis of the services in Section 3.1. The definition of complexity can be thought of as a large number of internal interactions between entities within piece of software, that could cause severe difficulties in either maintaining or modifying the software. Complexity can be further divided into accidental complexity and essential complexity. Accidental complexity pertains to problems software engineers create, due to suboptimal choice of technologies or lack of experience or knowledge, and can in theory be completely removed. While essential complexity consists of the problem to be solved, and thus can not be reduced, as it is a requirement to the software[97]. An analogy of complexity can be seen in Figure 3.9. Due to the combinations of architectures, hybrid architecture has increased complexity. This combination will lead to development of more entities, as well as more interactions between them, which again leads to more essential complexity.

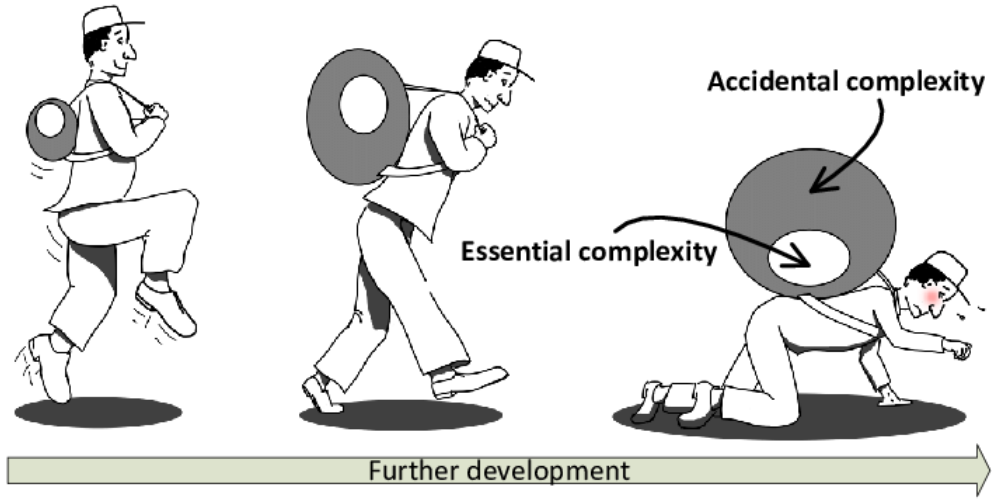


Figure 3.9: Analogy of complexity. As the system grows, the complexity of it increases in both essential and accidental complexity. Adapted from [98].

As much of Skypes architecture was based on the distributed super nodes, this made it really hard to maintain them in the case of failures. Mostly, this can be considered to be because of the way they had designed and created their system, which made it problematic when their a part of their network crashed, leading the entire service to crash. In essence accidental complexity caused issues for their architecture. While that helped to support the choice in changing architectures, the main reasons were due to the development of mobile devices. In order to properly support mobile devices, there would be several new distinct problems that would need solving, compared to a regular

PC or laptop. As such, a change to a client-server architecture would help reduce the overall essential complexity of the problem for them.

Looking at Pando, they were an anomaly of all the services explored, as they actually helped reduce complexity for their customers. Due to them offering the P2P and client coordination, their customers would only have to connect their own systems to Pando. This was less complex than having to develop everything on their own. Pando did, however, note several issues with its service, some of which were poor optimization of routing between peers, which is why they created the P4P workgroup, not to mention users from the customers would complain that it hogged network resources. These issues were a combination of essential and accidental complexity.

Spotify, with their hybrid architecture, had a lot of essential complexity. In order to provide a seamless streaming through a combination of peers and servers, they had several challenges. Their three levels of distribution meant great scalability, but it also meant more essential complexity. This was due to the three sources of streaming, which added some complexity of how to do this source selection process in an optimal way. Although Spotify moved away from hybrid architecture, their choice was also motivated by other reasons. Albeit they also noted that some features would in turn be easier to implement in a non-hybrid architecture.

In the case of Xunlei Kankan, they had a larger degree of essential complexity than the prior service, due to the naturally higher difficulty of video streaming compared to music streaming. Although their solution was similar to Spotify in how they too have “levels”, they also have more separation between and inside of these levels. For example, Xunlei developed secondary CDNs which would specifically handle initial streaming. This would make sure the user would have a seamless experience when connections to both networks were established and decided upon.

Of all the services covered in Section 3.1, none has more essential complexity than SLIVER.tv. Not only are they delivering live streaming in high definition, but they are also delivering it in 360° format. This, in combination with a desire to utilize peers for more than distribution, leads to more development and complexity than any other service that has been covered. In addition to this, they are also using blockchain as one of their core technologies, which will only add to the complexity. It is prudent to note that SLIVER.tv has developed their service with modern technology. Something that could potentially reduce accidental complexity to such a degree that it offsets most of the essential complexity they have or will face in the future.

Through the examples presented above, it is evident that complexity is a property within these services. It is possible to see a trend in how complexity of a hybrid architecture has changed, thanks to development of technology through the years. Essential complexity has increased, due to increased requirements from the services utilizing a hybrid archi-

tecture. Accidental complexity has been reduced, due to better tools and technologies to build these services. Complexity will be discussed in Section 4.3.

3.2.4 Secure

A less noticeable property is security. The importance of security is determined by what users expect when they start utilizing a service, and how much they stand to lose in the case of a security breach on that service. A security flaw can manifest itself in different ways within a service. Malicious users, scam artists and credit card fraud are only one side of security. A theoretical example of a malicious user corrupting content that pass through it is shown in Figure 3.10. Another aspect can be poor encryption or inaccessibility of the service, which are more focused on the flaws inadvertently created by the developers of the service. Although these two sides are different, they both are based on the same principle: security.

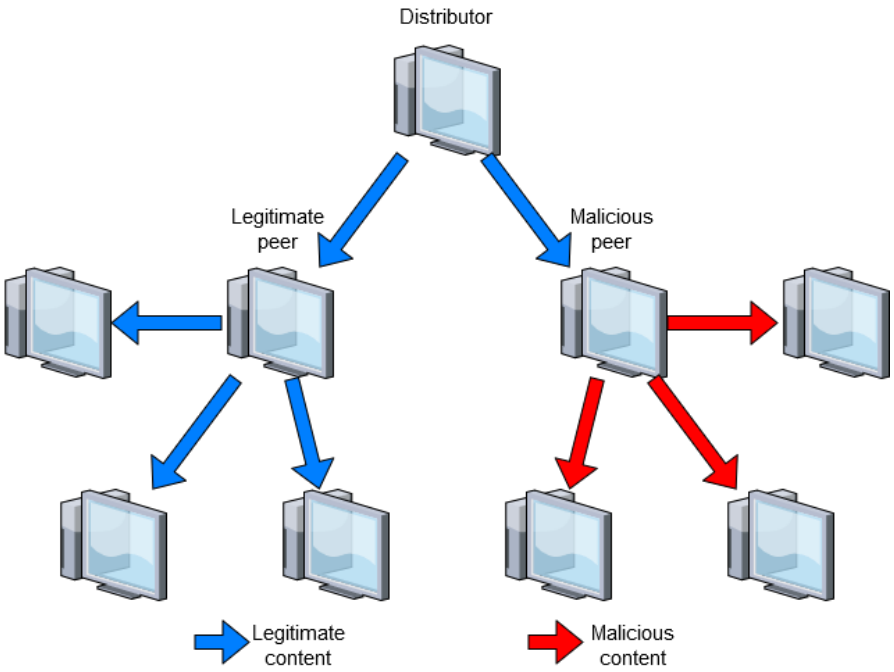


Figure 3.10: Theoretical example of a peer corrupting content that passes through it. A legitimate peer passes along the legitimate content, while a malicious peer corrupts the content before passing it along.

Skype allowed sharing of user-created content over instant messaging in a hybrid network, which meant that the service opened up possibilities of security risks from both users and architectural flaws. These problems got a larger focus when these flaws were

abused. As such, Skype struggled for a long time with problems that came from these security risks, which again came from how they had implemented a highly distributed hybrid architecture in their service.

One issue that plagued Skype was the demand from the service to access computers through firewalls, although they were not the only service that had this issue. Pando struggled with access through firewalls as well. Although Pando had this issue, they had a worse reputation of being a security hazard, due to their poor explanation of their service.

Spotify were more fortunate, as they did not experience the same level of security issues as Skype. Their architecture was made with a client that controlled what was shared through P2P, which verified and secured the content that was delivered. Due to the client securing the content shared, there were less instances of security flaws noticed.

Although sharing user-created content is mentioned as a security issue, it is not always the cause of security breaches. Xunlei Kankan had a breach in their security, due to developers in its parent company making a program that could infiltrate Android phones connected to an infected computer. Although the security breach came from the inside of the service, it highlighted a backdoor to the architecture, as well as revealing security concerns from users who had their machines infected.

The most prevalent of security flaws that plagued SLIVER.tv was not necessarily in their architecture, but rather in their service itself. The system became full of bots and fraudulent users, which abused the system to their advantage. Although this security issue is more pressing for the service itself, it also ruined the security of other legitimate users: they risked getting banned from the service, as it was hard to distinguish the legitimate from non-legitimate users.

Considering the prevalence of security issues that have been found throughout the services, security is a notable property of hybrid architecture. It should be noted that security issues can be very different from each other, and encompass different parts of a service, which might make it difficult to find. A discussion of security and its use in hybrid architecture will be conducted in Section 4.4.

3.2.5 Credible

The credibility of a software service is dependent on how they are presented towards users, and how they operate. A service that fail to maintain trust and respect from users, would realize that they are struggling to maintain the credibility they had. This combines with security in the fashion of poor security could result in loss of credibility.

The use of trust and respect as building blocks for credibility is an explanation with a theoretical view displayed in Figure 3.11

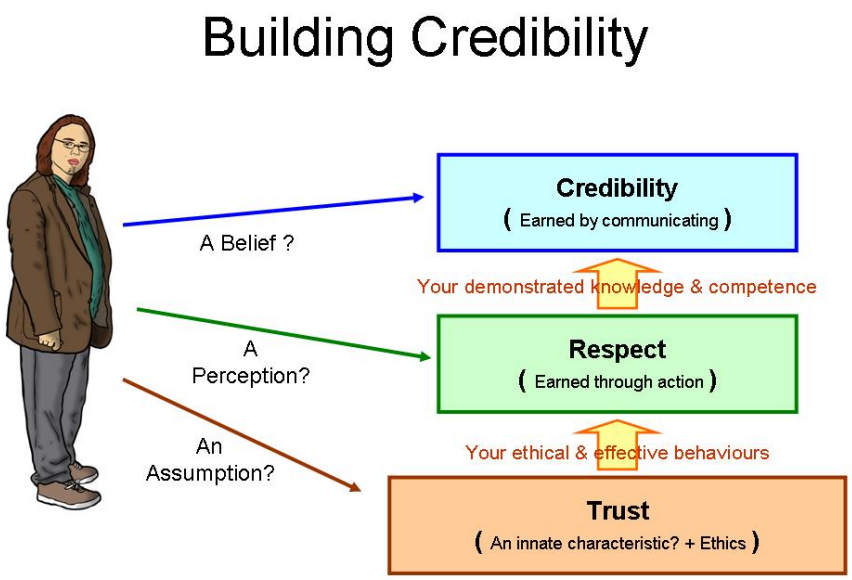


Figure 3.11: Theoretical building blocks for credibility. Trust and respect are needed to maintain credibility. Adapted from [99].

With the explanation of security flaws in Skype, they took a hit at their credibility. The issue of security flaws broke the trust, and privacy issues destroyed the respect users had in Skype, which lead to their credibility going downhill. Nonetheless, it was praised for the benefit it provided to telecommunication, and also got a boost to its credibility. This could also point towards that Skype did not lose all the credibility it had, since it was still in use.

Another example of loss of credibility is the story of Pando. The service itself did not suffer in the same way as Skype, since the service never went down due to architectural problems. However, the service lost respect due to poor communication that lead to users questioning the need to install Pando together with another service. Combining this with the fact that Pando used computer resources without telling the user, it broke down the respect and trust a user might have in the service.

On the other hand, Spotify embraced the need for trust and respect, in order to generate credibility. Although this was not always the case, as they had a lack of transparency in the use of P2P technology early on. Some Spotify users were negatively impacted by the use of P2P technology, and Spotify has since ensured transparency in such cases.

Due to being a platform that otherwise has functioned well, the trust and respect has been maintained, which in turn strengthened the credibility of the service.

Xunlei Kankan experienced an accidental loss of credibility that happened outside of their control. A security breach of the infrastructure from the parent company Xunlei, caused a loss of credibility. Due to the malware program being called Kankan, the same name as the Xunlei Kankan service, it impacted Xunlei Kankan's reputation. Although users may not have expressed a distaste towards the service because of this, the parent company gained credibility through their actions to reduce the spread of this breach.

The opposite happened with SLIVER.tv, where the service itself was responsible for the loss of credibility. When the service got rid of bots and fraudulent users, they also banned legitimate users from the service. Since the service also deals with virtual currency, which could potentially be transferred to rewards with real value, the ban of legitimate accounts were fairly unpopular.

Credibility of a service is valuable asset to the service providers and the users. Due to several cases where credibility have been covered, albeit either briefly or as part of security, it was selected as a represented property. Some cases will be elaborated and discussed in Section 4.5, to determine the importance of the property.

Chapter 4

Discussion

In this chapter, all properties that have been found in Section 3.2 will be discussed. The focus of this chapter will be a detailed discussion about the properties and whether or not they can be considered associated with a hybrid architecture. This is done by analyzing the impact a property has, with a focus on why a property is relevant and how it is relevant.

The first property that will be addressed is cost-effectivity, which is in Section 4.1. It is followed by scalability in Section 4.2. Complexity will be discussed in Section 4.3. After this, security is addressed in 4.4. Finally, credibility is covered in 4.5.

4.1 Cost-Effectivity

As seen in Chapter 3, there have been multiple references to how cost-effective a media streaming service could be when combining a typical client-server architecture and a P2P architecture. This has been seen in Spotify and Skype, which both utilized this hybrid architecture to reduce their costs in their startup phase. One of the benefits of this was that a startup may not have the appropriate funding to maintain servers and storage for the media. By relieving the storage and distribution with a P2P network, the funds saved could be utilized in different areas. These areas could include advertisements or other service-related expenses.

An important aspect on how a hybrid architecture can assist in reducing expenses comes from the sharing of resources between users that P2P is known for. In a study performed on Xunlei Kankan, it was found that 98% of a particular video had been de-

livered through peers. However, there are two points to consider when looking at those statistics.

The first point is if there are not enough peers, they may not be able to offload the centralized servers enough. In this case, relying on P2P to handle most of the distribution could be a problem. As such, it might be necessary to have good centralized servers to be able to secure proper content delivery. This is seen with SLIVER.tv, in regards to their choice in having centralized servers.

The second point refers to popularity of the content, with the differences between unpopular and popular content affecting which distribution method is viable. Popular content may have a large overlap in consumers that share content as peers, which make the 98% discussed above viable. On the other hand, unpopular content might need to be mainly distributed over an expensive centralized server, as there might not be enough peers.

But why is the centralized server expensive compared to sharing through peers? Take a hypothetical situation with a speculative ratio of streams between popular content versus unpopular content. There could be a million streams on the popular one, compared to every singular stream for the unpopular one. By making an assumption that both the popular content and the unpopular content are the same size, then the popular one will use more bandwidth.

To put it into perspective, Netflix requires 3 GB of bandwidth an hour for high definition streaming[100]. By assuming a continuous set of a million streams that are always streaming popular content, then it would be approximately 3 million GB worth of data that are streamed from Netflix per hour. On the other hand, there is a disparity of bandwidth usage when comparing those numbers to that of unpopular content, where there might only be a set of one stream that are always streaming. That would end up being 3 GB worth of data per hour. However, if as much as 98% of popular content was delivered through peers, the best case scenario for Netflix would be to only have to transmit 60 000 GB worth of data per hour. The difference of using mostly peers to stream video content versus server only is visualized in Figure 4.1.

This example looks at purely video streaming, which are quite large files. Looking at music streaming, streaming from Spotify at high quality uses 72 MB per hour[101]. A million streams of music would only yield approximately 72 000 GB worth of data in an hour. This means that music streaming which has smaller file sizes barely use more data with only a server, than video streaming using peers. Therefore, the file sizes actually matter when looking at a hybrid architecture to determine the cost-effectivity, as larger files have a greater benefit from utilizing peers. The difference between file sizes when streaming is visualized in Figure 4.2.

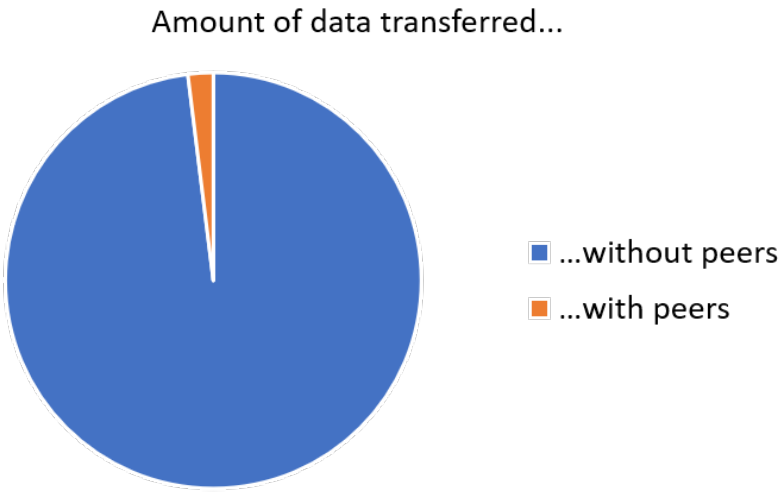


Figure 4.1: Visualization of difference between distributing only from server and distributing while peer assisted.

Although there is a benefit in utilizing peers, the question turns into whom it is beneficial for. In the best case scenario for Netflix that was mentioned earlier, it would only need to distribute about 60 000 GB worth of data from their servers, with the million peers having to distribute the rest. If Netflix uses 3 GB an hour for their video streams, and 98% are distributed through peers, then the peers would have to transmit on average 2,94 GB each. Due to the high percentage of peer-distributed data, the average bandwidth usage for the average peer would almost double with the use of a hybrid architecture. As such, the bandwidth cost of distributing content is not gone, it is just pushed from the company over to the consumer.

Despite that, the expenses would go down for video streaming services, due to the massive difference in data that is shown in Figure 4.1. Although, this architecture might not be as cost-effective for smaller file sizes. Not only are the total size difference vastly different, but there is also the need to maintain two different types of architectures. The savings of small file sizes distributed through a hybrid architecture might not warrant the extra development and maintenance cost that the combination adds.

It is also not only distributing that peers may be responsible for, but also temporary storage or computational power. This could turn problematic if the user is not expecting to share resources. It does not mean that everyone would be negative to the idea of sharing their resources, but in some cases they could be, especially if they are paying for the service. Additionally, the user could have a limit on their bandwidth usage, there could be hardware limits, or there could be battery usage problems.

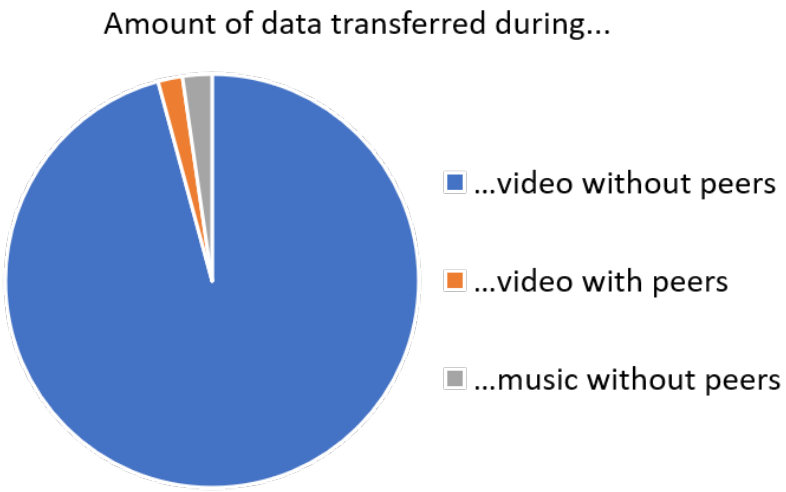


Figure 4.2: Visualization of streaming when the file sizes are very different.

These problems have been highlighted through Pando and Skype. In Pando's case, they did not notify users of their activity, and Skype created huge security risks by opening backdoors to public networks. Overall, the problem here seemed to be about not giving enough information to users. Thus, it is important that users are notified that they might be used as a peering node. They should also be offered the opportunity to easily turn it off, in case they do not have the possibility or desire to function as a peer.

However, if there is an option to turn the P2P functionality off to satisfy the users, it may become an issue if too many users disable it. This issue will then reduce the cost-effectivity of the service, due to less sharing between peers. A solution to this problem could be to provide the users that act as a peering node some form of compensation. This compensation might be priority on the service, or better quality on the media they want to stream. Another form of compensation is the one being utilized by SLIVER.tv, which is in the form of virtual currency. This currency can be used by the user to donate to a streamer of their choice, or can spend it on virtual items for themselves.

A different approach to the subject that will reduce the resources every user has to provide, is to utilize a cloud platform. Not only will this mitigate many of the issues related to dependencies of user's resources, it will also provide a cheaper alternative to expensive servers. As many services are hosted within its infrastructure, they can share the costs related to it. Furthermore, due to the infrastructure already being in place, startups might save some funds by utilizing cloud platforms, as they do not have to spend money on buying and setting up servers[22, 68, 102].

However, there has been worries of the hidden costs of cloud. For example, inexperience with developing for the cloud might cause overuse or poor utilization of the functionality offered by the cloud platform, meaning more expenses. Additionally, if the service does not know what functionality it needs from the cloud, they might end up paying more for functionality that is not needed. Still, cloud is comparatively cheaper than the service hosting the infrastructure itself[102].

To summarize, a hybrid architecture has the potential to be cost-effective. It does, however, rely on several points. For the service to be cost-effective, some of the costs are partially pushed over to the users. Following this, users might get irritated by an unexpected cost of using that service. It might also require a high amount of resources from users, which might impact their experience. One option is to have a transparent policy, in which the users know what is happening, and with a possible opt-out solution in case they do not want to be a part of it. This is also important as a hybrid architecture is very cost-effective when the shared file sizes are large, which also puts a heavy strain on users. Another option is to compensate the users for their shared resources, giving them a reason to keep using the service. This could be in the form of less advertisements, otherwise free usage of the service, or a payment in the form of virtual currency. A possible alternative to reduce costs is to utilize a cloud platform, which also would be beneficial to users. Overall, hybrid architecture has the potential to reduce the load on the centralized servers to a large degree, thereby saving the cost of more and larger servers.

4.2 Scalability

The second property that was identified was scalability. This can be seen in services such as Spotify and Pando, where both used hybrid architecture to handle loads past their own capacity and to ensure performance. This was successful for these services due to the spread of users, which impacted how the service operated. These services experienced that user activity could fluctuate based on geographical spread, day of the week and time of day. Another influence was the large overlap in what was commonly consumed. Thus, it was important that these services would be able to handle sudden spikes in activity at peak hours.

If these services could not handle the load, then the effects could be that all users would have to experience subpar performance, since the system could not keep up with demand. This was the case for services that delivered media, which enabled Pando to become a valuable resource in file delivery. Another way to ensure that the performance was acceptable was to do what Spotify did when they were just starting out, enact

a queuing system. This queuing system could control user growth to ensure that the service would not exceed the capacity of their servers.

While there is a concern of exceeding capacity, a lot of developments have helped in solving this problem. One such development is cloud platforms. These platforms have abundant of resources available, which is commonly higher than any requirement the service might have. Furthermore, virtualization technology used in the cloud will automatically scale to meet the needs for the service. Virtualization technology is an example of vertical scaling, which can be seen in Figure 4.3.

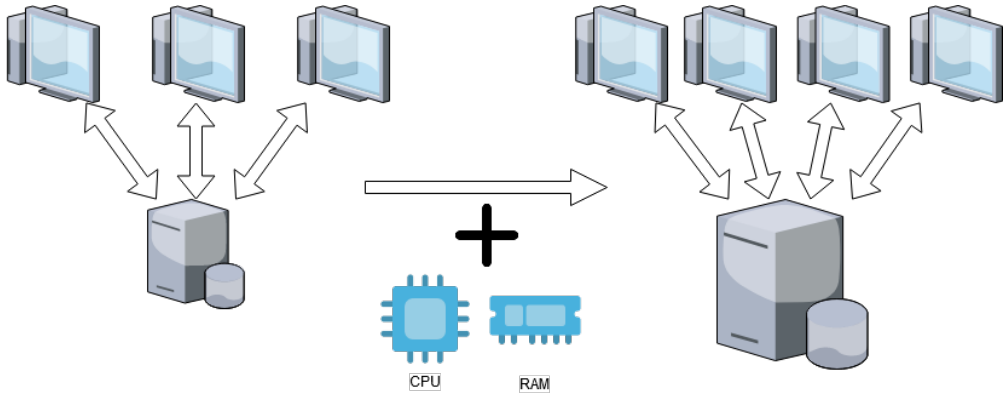


Figure 4.3: Example of vertical scaling, where resources are added to a node allowing more concurrent clients.

However, it is not uncommon that a cloud platform hosts multiple different services within its own infrastructure. Furthermore, while cloud providers usually plan to have more resources available than these services regularly use, there might still be unexpected situations. For example, if all the services hosted on the cloud concurrently had peak activity, the cloud infrastructure could potentially be unable to handle this load. In the case of use over or outside of the subscription plan, the cloud provider might limit the service performance, which would have a severe impact on scalability.

As can be seen, the scalability of dedicated servers and cloud can be somewhat varying, depending on several factors. However, thanks to a hybrid architecture, P2P can assist load balancing in such a case. As users join the service, they in turn help hosting the service by providing their own resources to help distribute content. Thanks to users becoming nodes, the service will be assisted due to horizontal scaling, as more and more users join the service. This helps negate the problems around potential resource and user limits around cloud, as well as potential cost problems that were discussed in Section 4.1. Furthermore, while servers has limits to how far they can be scaled, P2P on the other hand could potentially scale indefinitely as peers joined and supplied their own resources, as can be seen in Figure 4.4.

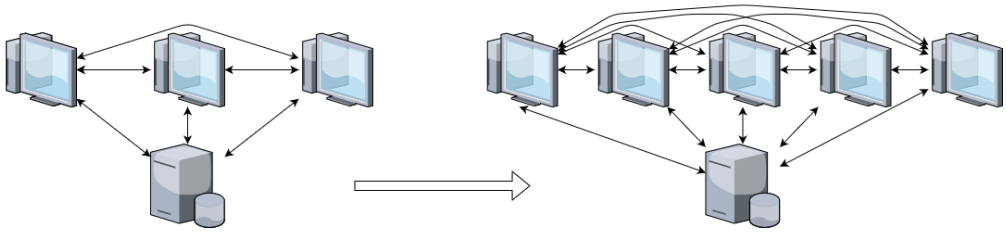


Figure 4.4: Illustration of horizontal scaling, where new nodes are added to the system.

This way of increasing scalability is what Netflix might have needed, as they struggled with upkeep of the service after their popularity skyrocketed. Since establishing new server farms for a service is not only costly, but also time consuming, a scalability solution with hybrid architecture could be very useful. Although this may be only useful as a short term solution, the benefit could also come in the form of happy customers who would come back to the service.

However, there are also some issues that can affect scalability with P2P, such as peer heterogeneity. While there is equality in a P2P network, not all peers have the same amount of resources or capabilities. In the case that the P2P network gets flooded with users that can not be considered suitable peers, then other users who connect to these peers might have a poor experience. This, in turn, can potentially lead to poor scalability of the service.

Furthermore, peers might not be dependable, and as such they might disappear abruptly while another peer is connected to them. This could lead to poor scalability for the remaining user, as it might suddenly have to find new peers, or find servers to stream from. However, both of these problems can be somewhat mitigated by using prediction, and having a decent amount of peers available at all times. Not to mention, being able to fall back on servers in the case that there are not enough peers.

Scalability can also be assured through P2P by predicting and seeding content to users ahead of time. This, in turn, could lead to the network being able to scale really well if the content suddenly became popular. An example of such an situation could be when a new movie or an episode of a popular series are being released. The same technique could also be used to prepare for maintenance, by making sure peers take a larger role in the distribution, while the servers are being maintained.

Overall, scaling can be really good in a hybrid architecture. The client-server architecture ensures scalability in lower loads and can provide great vertical scaling, while P2P provides scalability at higher loads and can provide great horizontal scaling. However, there are a few aspects which are important to note when talking about scalability of a hybrid architecture, such as limits and problems on both the server and P2P network.

Although, in all likelihood, scalability will be provided due to the positive attributes of both architectures, as well as both architectures solving problems with the other architecture.

4.3 Complexity

As a result of the decision of using a hybrid architecture, services such as Spotify and Skype gained scalability. However, this also meant an increase in complexity. This might stem from the fact that they were utilizing a combination of architectures, and therefore had to plan for, develop, maintain and support both architectures in their solutions. As such, complexity is a probable reason behind why the hybrid architecture has seen less use than traditional architectures. While there are benefits to choosing a hybrid architecture, there will inevitably also be disadvantages by choosing such an architecture.

Due to both Skype and Spotify being early adopters of a hybrid architecture, they in all likelihood had a larger degree of accidental complexity than some of the other services that had been covered. This could have been due to a lack of available technology and knowledge surrounding hybrid architectures, as it had not seen much use before this. As such, the early adopters might not have been able to handle or solve the increasing complexity of the hybrid architecture in the service they developed.

Some of this complexity could be attributed to P2P technology and its peers. The unsure nature of the peers could create several issues around complexity, since in media streaming, peers would need to have enough resources and be available to be able to provide seamless streaming. This could lead to a need to develop complex algorithms to measure the potential usefulness and availability of peers. Furthermore, it could also lead to development of complex algorithms to handle dynamic switching of peers based on results of these measurements. In addition to this, the use of mobile devices for P2P could introduce new problems which would increase complexity.

As technology and knowledge in the field has been improved since that time, it is possible that the complexity Skype and Spotify faced has been severely reduced. One example of technology that has been developed is WebRTC. It enables native P2P support for media streaming in the web browser over multitude of devices, as can be seen in Figure 4.5. Services such as SLIVER.tv are already utilizing technologies such as this in their solutions. Thanks to these improvements, as well as further development through libraries, this could have the potential to further reduce complexity for services that desire to utilize a hybrid architecture.

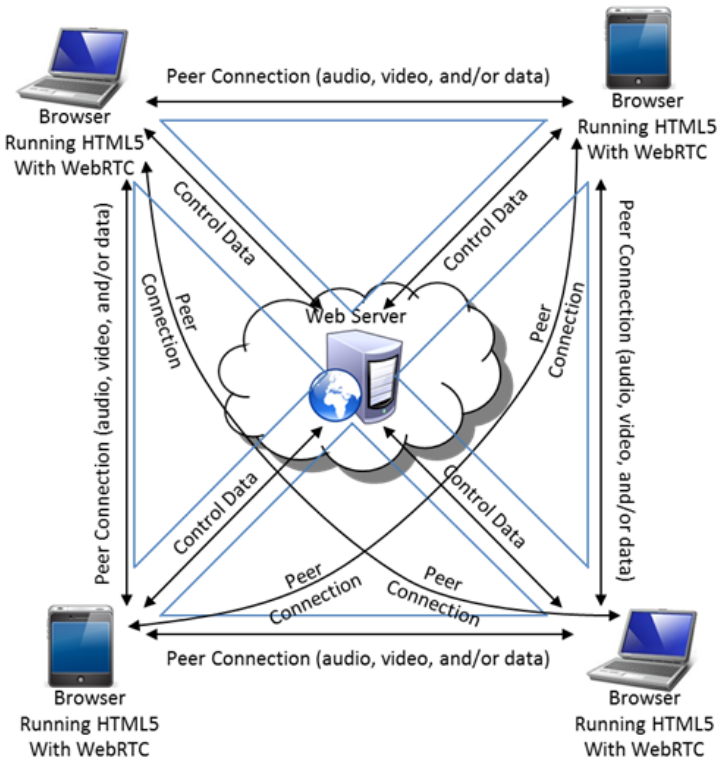


Figure 4.5: Example of how WebRTC enables communications between servers and peers. Adapted from [103].

Considering the usefulness of WebRTC on different devices, it could be an excellent technology for a service like YouTube, which already works on multiple different devices. Seeing that WebRTC already helps solve several of the complex issues with utilizing P2P for media streaming, this technology might be considered a critical factor if they were ever considering transitioning to a hybrid architecture. At the very least, it would reduce the work that was necessary to make a shift to a hybrid architecture for YouTube viable.

The emergence of cloud platforms is another development that has happened. As cloud platforms can solve or help handle several issues that a service would commonly have to face on its own, this might reduce the complexity of the service. This, in combination with functionality and tools that cloud platforms also bring to the table, has the potential to reduce complexity even further. Something that can be corroborated by both Skype and Spotify decisions, as they moved their own servers to the cloud at a later point.

However, while the cloud has the potential to reduce complexity, it might also increase it. The developer still has to ensure that the service works with the cloud, which requires configuration and integration of solutions that fit within the cloud platform. This, in turn, could mean an increase in complexity if the interfaces or development kits are not simple or intuitive enough. Not to mention that since the cloud platform is sometimes supplied by a third-party, the developers of the service might not have the access, permission or utilities to make changes that are necessary for the service. As such, developers of the service might have to utilize complex workarounds in order to solve problems they encounter with the cloud platform. Then again, these scenarios seem unlikely due to so many companies moving to the cloud, as well as cloud platforms often providing support teams for each service.

In relation to cloud services, Pando was quite similar to them. Although it was more in the way that they delivered a service, or rather a platform to their customers. By Pando enabling the use of hybrid architectures for services, the services did not have to worry too much about delivery or hosting aspects of their architecture. As such, they reduced the complexity for their customers, similarly to cloud platforms. This is backed up by how pleased Pando's customers were with the service. Therefore, it is not unlikely to think cloud platforms might start to develop their own services, offering something similar to what Pando did to their own customers in the future. This is corroborated by Microsoft's acquisition of Pando.

While it could be argued that some of these suggestions reduce complexity, is it not more correct to look at it as transferring complexity to other platforms or third-party developers? The service developers do not have to worry too much about the technical aspects of certain technologies, thus reducing complexity for themselves. However, the developers of the cloud platform or other technologies still have to handle the complexity of what they are providing to the service. The complexity does not really disappear, it is just shifted over to other third-party developers. However, it can be argued that it is beneficial to have third-party developers or platforms help in creating a better solution. Because the expertise they provide are skills, experience and knowledge, they can perhaps make less complex solution than these services could themselves.

To summarize, hybrid architectures increase the complexity of services due to the combination of both client-server and P2P architectures. Although there exists a few ways to simplify the architecture for the service, this commonly means shifting the complexity over to third-parties. This could create the illusion of reduced complexity, although that is not guaranteed. With technology and knowledge in the field being expanded, it might lead to reduced accidental complexity of hybrid architectures. This, in turn, could lead to developers with more technical expertise and experience. They could then create solutions based on proven methods and technologies, which would overall be less complex than the early experimental hybrid architectures.

4.4 Security

With increasingly complex services being developed using a hybrid architecture, there arose a necessity to discuss how the security of these services could be impacted by it. Through Chapter 3, concerns had been raised about potentially malicious peers abusing flaws or corrupting data in a hybrid architecture. This could negatively affect other peers, or even the service itself, as was the case with Skype.

Most of these security concerns can be attributed to the distributed nature of Skype. As Skype did not log or control messages being sent, it was considered extremely hard to discover the origin of an attack or a breach. In an attempt to fix these problems with Skype, developers released a buggy version of their software, which caused an accidental crash of their entire P2P network. This put the final nail in the coffin for its P2P part of the architecture, as they soon redesigned their architecture.

Skype's change in architecture also meant that they would remove the need to access computers through the firewall, which also was a security issue. However, Skype was not the only culprit in requiring access through the firewall, with Pando being another. These services essentially would open holes in the firewall, without requesting permission or notifying the users. Although this does not mean these services exploited their access, it created a security risk, since malicious files could be shared.

This might paint the picture that hybrid architectures are not very secure, however, it is also important to note that services like Skype were early adopters. As such, services like Skype might have trusted its own users to be reputable people, not utilizing their free service to do harm. Furthermore, Skype's security issues could potentially have been avoided if they had taken more control of their own P2P network, by creating security functionality to remotely detect and deal with malicious users.

On the other hand, Spotify was also early in utilizing a hybrid architecture with the primary function of media streaming. However, there was not much in regards to security flaws in their solution compared to Skype. A possibility is that Spotify delivered content they had stored in servers and thus could verify what was distributed between peers, while Skype did not store any content and could not verify what users distributed over their service. This then begs the question if hybrid architecture is the culprit for security issues, or if the addition of user-created content the real culprit.

A media streaming service that could be used to look at the security risk of user-created content is YouTube. The content on YouTube is mostly user-created, and as such, could theoretically be a victim of the same security flaws that plagued Skype. In spite of this, YouTube does not appear to have any issues regarding distribution of user-created content. This could mean that having a centralized server means more control over what users are doing on the service, and thus makes it easier to remove or prevent instigators

before any problems arise. This could point in the direction that having a centralized server that can verify content could be a necessity to ensure security.

Through this, it could be argued that a completely centralized service would be more secure. On one hand, more control could lead to more oversight of interactions and activity in a service. On the other hand, a vulnerability in a centralized service could give an attacker more control, and access to a larger part of the system than in a distributed system. Such vulnerabilities has to some degree been rectified in cloud computing, as users connect to the service through an client instance in the cloud, and do not directly communicate with the server that is also hosted in the cloud.

While a centralized solution could be more secure, an issue that has become apparent is how user privacy can be affected by centralized systems. As there have been cases of user data or activity having been shared with government agencies and other organizations. For example, it was revealed that after Skype had changed to a centralized solution, they had been sharing conversations with the American government. To make matters worse, they had not informed their users of this activity in any way, which breached their trust with the service. Although privacy concerns are still present in cloud computing, the argument is that since cloud platforms host a multitude of services from a multitude of locations, they need to maintain a neutral stance to accommodate their customers. Thus, the responsibility for any privacy breach lies solely on the service itself, and not on the cloud platform, as the service is the one that authorized the breach.

An solution to securing a distributed service could be to do what SLIVER.tv did, which is utilize blockchain technology. Due to the distributed nature of blockchain technology, a malicious user could potentially be discovered and, with the consent of a quorum of its peers, stopped. With every peers action being controlled and verified, this could potentially secure a distributed service from interference or malicious intent. Users that act maliciously can even be punished for their attempt, and potentially be removed from the system.

However, such a solution will likely require a large overhead of resources, negatively affecting performance of a system. Not to mention, if a quorum was controlled by malicious users, they could in essence manipulate the system at will. Nonetheless, such a scenario is very unlikely, as the quorum would consist of a random subset of several hundreds of thousands or even millions of users.

That said, maybe a quorum is the best option in a distributed architecture, since users can police themselves without oversight from the developers. As there have been cases where developers have been the ones to exploit security issues. For example, Xunlei Kankan had a issue where developers created a program which downloaded malware onto customers phones. SLIVER.tv struggled with rampant use of bots and fraudulent

users and banned more users than what they needed to. These two cases show that security issues is not exclusively related to users, as developers can be a part of security issues too.

While security is a concern to be noted in a hybrid architecture, it might not be as important for media streaming solutions, which is reflected in differences between Skype and Spotify concerning security issues. Concerns in such a case could be corrupted media files, which could probably be detected by the service or client through integrity checks. In addition, media streaming services would likely contain data that are less sensitive in response to privacy or security. In any case, the severity would likely be of a smaller scale than other types of services, such as services storing very sensitive data.

4.5 Credibility

In the case that security of a service has been breached, this can have severe consequences for the credibility of the service. For example, Skype got backlash due to how the backdoor could be used for malicious intent, especially in larger enterprise networks. Furthermore, the scandal involving Skype sharing conversations with intelligence agencies got them more backlash, as users felt their privacy had been breached without their consent.

However, these events did not impact user count that much, as it still had a majority in the market. On the other hand, several corporations moved their businesses away from Skype due to these issues. Therefore, it mostly impacted the use of Skype from businesses, with private use still retaining activity.

Although security issues impact credibility, there are also other factors that can affect it. Such as with Pando, where users were not notified of Pando being installed, as it was bundled with the service they were installing. With users suddenly experiencing problems due to Pando being installed, many assumed it was a virus leeching of their computer resources. While it was using computer resources, it was merely doing so in order to support the service they originally installed.

As such, it is quite understandable that users questioned the services credibility, when it was revealed that Pando was installed without notifying the users. While this impacted their credibility, the entire situation could have been entirely avoided if they had notified their users of the installation and function of Pando during installation of the service. Users might not mind sharing their resources, if they had known about Pando beforehand. Much of the backlash came from users who simply did not have the resources or capabilities to share with the service.

Although credibility is mentioned more often when it is lost instead of gained, there are ways to rectify situations to regain credibility. One such example was when Xunlei Kankan had a security breach, which impacted their credibility. Their answer was to quickly create an uninstaller that would remove all files related to the malicious program and inform their users. Such actions will regain credibility due to services acknowledging their mistakes and fixing them.

This is a problem that SLIVER.tv did not address properly. With their problems of bots and fraudulent users, their system banned many users, some of which were legitimate. When these users questioned why they were banned, and subsequently lost their accumulated virtual currency, SLIVER.tv did not give adequate responses. Since the developers did not provide proper dialogue or communication with the affected users, their credibility as a service dropped.

However, the issues with proper dialogue is not only a problem for media streaming services utilizing a hybrid architecture. For example, YouTube's artifical intelligence (AI) system banned several channels that were reporting on, or providing, evidence of human rights violations in Syria and other locations, due to it flagging many videos as disturbing or violent. As such, evidence of these atrocities were deleted, and possibly lost forever. Even when asked to restore the video due to it being used as evidence in a court case, YouTube denied, as they agreed with the AI's verdict. This caused a loss of credibility for YouTube, as they had previously claimed they supported users who uploaded evidence of human rights violations, and would continue offering tools to help them promote human rights[104].

Thanks to situations like this, it could be argued that transparency with users is key in a hybrid architecture. If a service does not communicate properly with its users, then the users will wonder about the credibility of the service. Equally important is to make sure that users have been informed, as well as given consent to sharing their computational resources with the service. This might help boost the trust between the user and the service, which can actually improve the credibility of the service.

Another issue that can impact credibility is reliability. If a service frequently have issues with uptime or has performance issues, users might not find the service reliable. For example, the case of the users in Spotify that could not function as peers, as they had too poor internet connection to stream and distribute content at the same time. Due to this case, Spotify lost some credibility. However, in some cases, a service might corner a market, where reliability and credibility does not impact the service too much. Skype is a good example of this, as even though they were impacted, they still retained much of their user base. On the other hand, there are services that have alternatives, like cloud platforms. If one cloud platform provider does not deliver on promises or functionality,

there are alternative cloud platforms. This means that a successful cloud platform has high credibility.

While credibility is a property that has some significance in regards to a hybrid architecture, it is not one of the most important ones. A main reason is that regardless of service and architecture, credibility is equally important for everyone and is not significant for hybrid architecture in particular. In regards to media streaming, as long as users are informed and give their consent, it is unlikely that it will be important to focus on as a service. This is due to the less sensitive data collected, as well as the lower impact security issues could potentially have in a media streaming service.

Chapter 5

Evaluation

This chapter will evaluate the properties that have been found in this thesis, and based on the discussion, deem if they are significant for media streaming services. This selection of significant properties will happen in Section 5.1. Based on knowledge from Chapter 4 and Section 5.1, the viability of hybrid architecture will be evaluated in Section 5.2. This is supplemented by aspects to consider when using hybrid architecture, which is presented in Section 5.3. Finally, a critical evaluation of the research will be conducted in Section 5.4.

5.1 Significant Properties

Several properties were found in the services utilizing a hybrid architecture. Some of the properties were found to be associated with hybrid architecture to a certain degree, while others were more separate. Then again, only a few of the properties were found to be significant in relation to media streaming services. While a short explanation behind the reasoning will be conducted, Chapter 4 has a more thorough coverage of each property and reasoning.

One of the most prominent properties discovered was *cost-effectivity*. Due to the utilization of user resources at times of high load in combination with the utilization of servers at lower loads, a hybrid architecture managed to be quite cost-effective. Considering that media streaming has varying loads, in combination with commonly large costs related to the process of hosting and streaming, it is a property that is significant in relation to media streaming services.

Another property that was found is *scalability*. *Scalability* also came from the utilization of user resources during higher load, as well as utilization of servers during lower loads. In relation to media streaming, the property is significant due to the sudden spikes in user activity that can occur on events such as new releases or regular fluctuations in user activity. Seeing that these spikes can overburden the server in a regular client-server architecture, the assistance of peers can be a significant improvement to scalability. Furthermore, as P2P has poor scalability at lower peer counts, servers can be used to ensure scalability during these periods of low activity by ensuring consistent performance of media streaming.

The third property that was found to exist within hybrid architectures was *complexity*. It came from the fact that developers had to face challenges of both architectures, while combining them into one. Not to mention the point that it still had to meet the requirements set upon the service. In regards to media streaming, it was significant due to potential problems this could cause with the performance or availability of the service, as complexity could cripple the maintainability. This, in turn, could lead to poor user experience and the service failing to meet expectations. Not to mention that media streaming in itself is a complex problem due to the wast amount of potential users.

Security was another property that was discovered in relation to hybrid architectures. This was due to the distributed nature and user participation that came with hybrid architectures. However, it was considered less important in relation to media streaming. Media streaming work with little sensitive personal data, in addition to the lower impact that a security breach could potentially have within a media streaming solution. Additionally, in some cases of media streaming services, there are limited possibility to change files and force a security breach. Although, if a security breach takes down the service, a large number of users could be affected by it.

The final property that had been discovered was *credibility*. Due to the potential issues with *security*, it became noted as something that might be impacted by a breach, and the fallout of said breach. While trust between user and service was considered important in a hybrid architecture, it is not less important in other architectures either. This means that even though *credibility* is important, it is not uniquely significant for hybrid architectures with focus on media streaming.

5.2 Viability of Hybrid Architecture in Media Streaming

Throughout Chapter 3.1, several services have been covered. All of which have implemented a hybrid architecture to different degrees of success. Furthermore, the properties

discovered have been analyzed and discussed in detail in Chapter 4, and summarized in Section 5.1. Still, none of these have discussed the actual viability of a hybrid architecture for media streaming, which will be covered in this section.

Seeing how most of the services that have been covered in this thesis either have closed down or transitioned away from a hybrid architecture, one could make the argument that a hybrid architecture has poor viability. On one hand, this is supported by the problems these services have had due to their choice of architecture. On the other hand, there are reasons these services switched away that are not related to the viability of hybrid architecture.

One such example is Spotify, where they reasoned that they had built up a good enough consumer base to justify the change to client-server. They also had a few user complaints about bandwidth usage, which Spotify felt was an unnecessary grievance for their users. This, in combination with the reduced costs of server hardware and bandwidth, motivated their change, more than any potential issues they might have faced if had they not changed. Not to mention, Spotify was a startup when they used the architecture, as such the cost-effectivity they gained from a hybrid might have helped them survive to this point.

However, on the other side of the spectrum is Skype. It changed away from the hybrid architecture due to viability issues. The crash of their network, without their ability to do remote maintenance was a blow to their service. This issue was grave, considering it was one of the largest internet telecommunications applications at that time. Another relevant issue was that nodes that relayed messages could intercept said messages and replace them, which also became a problem for Skype.

While this weakens the viability case of a hybrid architecture, it is also important to note that Skype was one of the earliest adopters of a hybrid architecture. Considering the service was first released in 2003, with much of it being based on complex proprietary technology and knowledge, it was impressive that the service worked fine for such a long time. On this basis, it can be argued that Skype managed to create a viable service, even when the current technology and knowledge would argue it was not viable. Perhaps it even helped pave the way for future services that utilized a hybrid architecture. Thanks in part due to pushing technology and knowledge forward, making it more viable.

As such, one of the greatest examples of how technology and knowledge in the field has moved forward is SLIVER.tv. Their live streaming service utilize modern technologies such as blockchain and WebRTC, in order to organize workload and distribute the media content. This has helped solve many different issues that previous services have faced in their development of a hybrid architecture. Thus it strengthens the argument that hybrid architectures are viable, and have become more viable for media streaming. Furthermore, SLIVER.tv stated that the more common client-server architecture did not

meet the requirements they had for their service. Something that can be explained due to SLIVER.tv's larger data sizes, in addition to the requirement of lower latencies. This also provides evidence to the viability of a hybrid architecture in media streaming.

These arguments, in combination with the properties that has been discussed in Chapter 4 and Section 5.1, provides evidence of the viability of a hybrid architecture within the realm of media streaming. While there are examples of services using and moving away from it, these services were commonly startups that needed to establish, or gain a share in the market. As such, it is likely that a hybrid architecture has the highest viability in a startup within media streaming. Another example of where a hybrid architecture might be a viable option is in the case of a service needing to reach a minimum required user base to support a switch to pure P2P architecture. There are some points that are important to note before deciding to use a hybrid architecture in the realm of media streaming. These points impact the viability of a hybrid architecture, and will be covered in Subsection 5.3.

5.3 Points to Consider When Deciding to Use a Hybrid Architecture

This subsection will cover some of the important points to note before deciding to use a hybrid architecture. The points can have a severe impact on the viability, depending on how the service works and what it delivers. It also contains potential solutions to issues that have been highlighted throughout this paper. It is mainly focused on media streaming services, but some points might be relevant to note for other services that desire to utilize a hybrid architecture.

5.3.1 User Base

A suitable user base should be available to make sure a hybrid architecture will be viable. If the service only has a small potential user base, this could make the entire P2P network inefficient. As such, it is important to note a minimal viable number of peers required to be able to utilize the P2P network to a satisfactory degree. The more resources a service needs, the higher the probability is that more peers will be required. Not to mention a heavier burden will likely be placed on the peers.

5.3.2 Resource Usage & Overlap

In order for a hybrid architecture to be viable, there has to be a large enough requirement for resources. For example, larger data sizes will mean more usage of memory, as well as bandwidth for transmission. Thus, a service doing video streaming likely will have larger resource requirements comparatively to a service that only streams music. Therefore, video streaming might be more viable when using a hybrid architecture. Furthermore, there has to be enough overlap into what content users consume. If the users have unique tastes, and little to no overlap in what content they consume, in theory the P2P network will be almost completely useless.

5.3.3 Transparency & Voluntary

The users should be clearly notified that their resources could be used in a P2P network. This is due to some users having constraints or rules that, when broken, could negatively impact their personal lives. For example, corporate networks might have rules in place that any service utilizing P2P is banned within the network, something that could potentially lead to the user being penalized by their employer. Another example is that some users have limits to how much resources they have available and can use during a period of time, which might negatively impact their experience of the service. The resources could be a cellular data plan or even battery life, and as such, not being informed could lead to unforeseen expenses or other problems for the user.

In relation to being notified, it is also a good idea to allow the participation in the P2P network to be entirely voluntary. As such, users with these constraints or rules should have the possibility to opt out of participation, instead of being unable to use or having to limit their use of the service. If there is a worry that too many users will opt out, a solution could be to incentivize users. For example, by prioritizing the users that participate, or grant them better quality on their service. Other incentives could be virtual currency or points that can be spent within the service in some way, or even access to improved functionality or early access to new content.

5.3.4 Capability

Due to the high complexity of creating a service that utilizes a hybrid architecture, it is important to note the capabilities of both the software engineers and current technology. While a hybrid architecture has several benefits, these benefits can be squandered in the case that the developers or technology is not up to par to handle the requirements of the service. Thus, even if a working service has been developed, it might not be

competitive enough in regards to quality compared to other similar services. This could mean users would never return after having tried the service. As such, the inherently higher complexity that comes with hybrid architectures is important to note.

5.4 Research Critique

The first point of critique is the selection process of services. Some of the services selected were found to be less suited for this type of research than the other services. As some of the services that were selected are closed source, this meant available documentation was limited. Furthermore, some of the services are rather old or have not used the hybrid architecture for many years, and as such, documentation might have disappeared or is contradictory due to changes within the service. These elements might have impacted the accuracy of the descriptions of the services, due to the limited information available.

This then leads to the second point of critique, the document selection. Since document analysis was the main data generation method, the thesis is dependent on a multitude of documents from a myriad of sources. As such, there is likely a great disparity in quality and objectivity between these documents. Even if the focus was to mainly utilize technical documents and research papers, in some cases it was necessary to use other documents, such as blog posts, news articles or user comments, to fill gaps in information. This might have impacted the credibility of the results of this thesis, as not all of these can be considered objective and credible sources.

Furthermore, since the thesis was dependent on documents, another critique is the credibility of the authors of those documents. As those documents might very well have been impacted by research bias to some degree, which could have repercussion for this research. Not to mention if the documents were authored by a third-party, they might not have had the necessary access or knowledge to provide accurate results. As such, any inaccuracy or fault might have inadvertently been included in this thesis, potentially affecting the correctness of the results.

Finally, the last critique is that the focus on media streaming might have impacted the discovery of properties in a hybrid architecture in general. As such, it is possible that other important properties in regards to a hybrid architecture have been neglected or has lost focus. Furthermore, other properties may have gotten more focus than warranted. While important to consider as critique, it is equally important to note as a limitation for the thesis.

Chapter 6

Conclusion

This chapter will conclude the thesis and provide a short, concrete answer to the research questions. In addition, this chapter will also include the contributions of this thesis, which is in Section 6.1. Finally, future work will be discussed in Section 6.2.

Regarding the question of what properties can be found in services using a hybrid architecture, a few properties have been identified. These properties are cost-effectivity, scalability, complexity, security and credibility. These properties were identified because of their prevalency in documents about the selected services. Due to related works, some properties were expected, such as cost-effectivity and scalability.

Concerning which of these properties are significant in relation to media streaming, the five identified properties are reduced to only three properties that can be considered significant. The properties that are significant are cost-effectivity, scalability and complexity. Since the related works also pointed towards cost-effectivity and scalability, the significance of those two properties are considered valid.

Finally, regarding the question of viability of a hybrid architecture in a media streaming service: a hybrid architecture can be viable in a media streaming service, but several factors need to be considered. The cost-effectivity provided by a hybrid architecture is significant, but the costs may be pushed on to the users. The scalable nature of hybrid architecture can be helpful for alleviating the server in peak periods, but it may be hard to fix if a node in the system becomes corrupt or malicious. A service can utilize the best of both architectures, but it can become a quite complex service, with the need to maintain and control a vastly complicated system.

So the viability of hybrid architecture is present, with the option of utilizing it for both existing and new media streaming services. There have been examples of hybrid archi-

tecture failing, and there have been examples of hybrid architecture succeeding. However, a majority of the services covered here is not currently using hybrid architecture, having instead switched to using cloud computing.

There are definitely negative aspects to using cloud computing, but there are also great benefits. An aspect that has not been considered in this thesis is whether or not a certain number of users would make using cloud computing more viable, or if below that threshold, other options are better. With such arguments to consider, it is possible to consider hybrid architecture over cloud computing. Then again, cloud computing provides benefits in such a scale that it might be hard to consider other architectures and solutions.

To conclude, hybrid architecture is viable for media streaming services, although it might not always be the best alternative.

6.1 Contributions

The main contribution of this thesis is to illuminate potential advantages and disadvantages with hybrid architectures, in order to assist decision-making in the choice of a system architecture. This is achieved by providing a new perspective on preexisting theories and results, as well as new results to the field. The significance of the contributions is how they are based on several real-life services, instead of theoretical or simulated systems.

6.2 Future Work

One thing that became apparent during the research is that almost all of the services that were explored, either moved away from, or are planning to move away from, a hybrid architecture in the future. Thus, it would be interesting to do a more in-depth investigation as to why services do not seem to persist in their use of a hybrid architecture. This could help in determining whether or not a hybrid architecture has viability as a prolonged solution.

Furthermore, while this thesis found some properties that were significant for hybrid architectures within media streaming, there could be other significant properties that this thesis did not discover. For example, this thesis noted properties such as availability and maintainability shortly, as aspects of the other properties. This happened due to a lack of information from the documents to justify including it as a separate property in this thesis. There might even be other services or documentation available, that was not

found in relation to this study, that could provide new and interesting information. As such, a further study into other properties might be useful to gain further insights.

In relation to this, it might also be interesting to conduct an experiment with a simulated media streaming service, utilizing a hybrid architecture. An example of a test case could be streaming a media file, where the number of peers are a changeable parameter. This could then be expanded further, by changing the type of media from sound to video, or even change the size and quality of the media file, to test out how different content impact distribution from server and peers. Through this, insights into the properties of a hybrid architecture in media streaming could be gained, and the actual impact of a hybrid architecture in media streaming could be tested. The data collected could be used to figure out how cost-effective a hybrid architecture could be compared to a client-server architecture, or even provide a measure of minimal number of viable peers for different types of content.

Another thing that was discovered and deemed interesting in regards to this thesis, is the possibility of cloud platforms offering P2P as a service. Since many of the covered services in this thesis transitioned to the cloud, it could be an interesting solution to allow cloud platforms to handle P2P on their behalf, in the same way that cloud is currently handling the servers. This could have major significance in relation to the viability of hybrid architectures, as well as possibilities for cloud computing.

Although cloud computing has been mentioned throughout this thesis, it is not the only technology that needs to be considered. SLIVER.tv used blockchain to increase the capabilities of hybrid architecture for streaming media content. Since SLIVER.tv is a fairly recent service, it might be useful to research improvements to the technologies it uses, especially with the focus on viability. It might also be interesting to have a further look into how blockchain technology can be used in relation to a hybrid architecture. This could in turn lead towards a better version of hybrid architecture, using modern technological aspects.

Finally, while not mentioned widely in this thesis, the impact hybrid architectures have on society and the environment might be interesting to study. The partially distributed nature could potentially support user privacy and freedom, as it would be harder for governments or organizations to monitor network traffic or activity. At the same time, the distributed architecture might utilize devices that are less optimized for server related tasks, like smartphones, and as such, may use more power than data centers that are optimized for such tasks.

Bibliography

[1] *Internet consumer data traffic worldwide by segment 2016-2021*. June 2017. URL: https://www.statista.com/statistics/454951/mobile-data-traffic-worldwide-by-application-category (Accessed 8. Nov. 2018).

[2] *Global number of YouTube viewers 2016-2021*. Feb. 2018. URL: https://www.statista.com/statistics/805656/number-youtube-viewers-worldwide (Accessed 31. Oct. 2018).

[3] Felix Richter. *Infographic: Netflix Continues to Grow Internationally*. Oct. 2018. URL: https://www.statista.com/chart/10311/netflix-subscriptions-usa-international (Accessed 31. Oct. 2018).

[4] 'You know what's cool? A billion hours'. In: *Official YouTube Blog* (Feb. 2017). URL: https://youtube.googleblog.com/2017/02/you-know-whats-cool-billion-hours.html (Accessed 30. Oct. 2018).

[5] Christine Gallup. 'How Much Data Does YouTube Use?' In: *WhistleOut* (Nov. 2017). URL: https://www.whistleout.com.au/MobilePhones/Guides/How-Much-Data-Does-YouTube-Use (Accessed 30. Oct. 2018).

[6] Isla McKetta. 'The World's Internet Speeds Increased More than 30% in 2017. Are You Keeping Up?' In: *Speedtest Stories & Analysis: Data-driven articles on internet speeds* (Dec. 2017). URL: http://www.speedtest.net/insights/blog/global-speed-2017 (Accessed 13. Nov. 2018).

[7] Robert R. Schaller. 'Moore's law: past, present and future'. In: *IEEE Spectrum* 34.6 (June 1997), pp. 52–59. ISSN: 0018-9235. DOI: 10.1109/6.591665.

[8] 'Decentralized video streaming, powered by users and an innovative new blockchain version 2.0'. In: *Theta Network* (Nov. 2018). URL: https://s3.us-east-2.amazonaws.com/assets.thetatoken.org/Theta-

white-paper-latest.pdf?v=1547722765.743 (Accessed 13. Nov. 2018).

[9] Ira S. Rubinstein, Gregory T. Nojeim and Ronald D. Lee. 'Systematic government access to personal data: a comparative analysis†'. In: *International Data Privacy Law* 4.2 (May 2014), pp. 96–119. DOI: 10.1093/idpl/ipu004. URL: http://dx.doi.org/10.1093/idpl/ipu004.

[10] Asunción Esteve. 'The business of personal data: Google, Facebook, and privacy issues in the EU and the USA'. In: *International Data Privacy Law* 7.1 (Mar. 2017), pp. 36–47. DOI: 10.1093/idpl/ipw026. URL: http://dx.doi.org/10.1093/idpl/ipw026.

[11] Anirban Mondal and Masaru Kitsuregawa. 'Privacy, Security and Trust in P2P environments: A Perspective'. In: *17th International Workshop on Database and Expert Systems Applications (DEXA'06)*. Sept. 2006, pp. 682–686. DOI: 10.1109/DEXA.2006.116.

[12] George Reese. *Database Programming with JDBC and Java, Second Edition*. Ed. by Andy Oram. 2nd. Sebastopol, CA, USA: O'Reilly & Associates, Inc., 2000, pp. 126–136. ISBN: 1565926161.

[13] R. Schollmeier. 'A definition of peer-to-peer networking for the classification of peer-to-peer architectures and applications'. In: *Proceedings First International Conference on Peer-to-Peer Computing*. Sept. 2001, pp. 101–102. DOI: 10.1109/P2P.2001.990434.

[14] HMN Dilum Bandara and Anura P Jayasumana. 'Collaborative applications over peer-to-peer systems–challenges and solutions'. In: *Peer-to-Peer Networking and Applications* 6.3 (2013), pp. 257–276.

[15] Marco Danelutto, Paraskevi Fragopoulou and Vladimir Getov. *Making Grids Work: Proceedings of the CoreGRID Workshop on Programming Models Grid and P2P System Architecture Grid Systems, Tools and Environments 12-13 June 2007, Heraklion, Crete, Greece*. Vol. 7. Springer Science & Business Media, 2008.

[16] Qin Lv, Sylvia Ratnasamy and Scott Shenker. 'Can heterogeneity make gnutella scalable?' In: *International Workshop on Peer-to-Peer Systems*. Springer. 2002, pp. 94–103.

[17] M. Kelaskar et al. 'A Study of Discovery Mechanisms for Peer-to-Peer Applications'. In: *2nd IEEE/ACM International Symposium on Cluster Computing and the Grid (CCGRID'02)*. May 2002, pp. 444–444. DOI: 10.1109/CCGRID.2002.1017187.

[18] *How Content Delivery Networks Work*. Apr. 2015. URL: https://www.cdnetworks.com/en/news/how-content-delivery-networks-work/4258 (Accessed 3. Dec. 2018).

[19] *Company History | Akamai*. Apr. 2019. URL: https://www.akamai.com/uk/en/about/company-history.jsp (Accessed 29. Apr. 2019).

[20] Nicholas C. Zakas. *How content delivery networks (CDNs) work*. Nov. 2011. URL: https://humanwhocodes.com/blog/2011/11/29/how-content-delivery-networks-cdns-work (Accessed 29. Apr. 2019).

[21] VideoCoin. 'Three Ways CDNs Have Changed Since Akamai's First Content Delivery Network'. In: *Medium* (Apr. 2018). URL: https://medium.com/videocoin/three-ways-cdns-have-changed-since-akamais-first-content-delivery-network-7c50c1dfb05c (Accessed 29. Apr. 2019).

[22] Eric Knorr. 'What is cloud computing? Everything you need to know now'. In: *InfoWorld* (Oct. 2018). URL: https://www.infoworld.com/article/2683784/what-is-cloud-computing.html (Accessed 6. Mar. 2019).

[23] Yuping Xing and Yongzhao Zhan. 'Virtualization and Cloud Computing'. In: *Future Wireless Networks and Information Systems*. Ed. by Ying Zhang. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 305–312. ISBN: 978-3-642-27323-0.

[24] Liu Murong. 'Overview and Appliance of Some Streaming Media Software Solutions'. In: (2009). URL: http://www.theseus.fi/handle/10024/57742.

[25] Dongyan Xu et al. 'Analysis of a CDN-P2P hybrid architecture for cost-effective streaming media distribution'. In: *Multimedia Systems* 11.4 (Apr. 2006), pp. 383–399. ISSN: 1432-1882. DOI: 10.1007/s00530-006-0015-3.

[26] Debjani Ghosh, Payas Rajan and Mayank Pandey. 'P2P-VoD Streaming'. In: *Advanced Computing, Networking and Informatics-Volume 2*. Springer, 2014, pp. 169–180.

[27] Tom Macaulay. *10 Years On: How Netflix completed a historic cloud migration with AWS*. Sept. 2018. URL: https://www.computerworlduk.com/cloud-computing/how-netflix-moved-cloud-become-global-internet-tv-network-3683479 (Accessed 3. Apr. 2019).

[28] Richard Thelwell. *Why did Netflix migrate to the AWS Cloud?* Mar. 2016. URL: https://www.matillion.com/blog/redshift/why-did-netflix-migrate-to-the-aws-cloud (Accessed 24. Apr. 2019).

[29] Cam Cullen. *Sandvine releases 2019 Mobile Internet Phenomena Report*. Feb. 2019. URL: https://www.sandvine.com/press-releases/sandvine-releases-2019-mobile-internet-phenomena-report (Accessed 24. Apr. 2019).

[30] Salman A. Baset and Henning G. Schulzrinne. 'An Analysis of the Skype Peer-to-Peer Internet Telephony Protocol'. In: Columbia University, Department of Computer Science, 2005. DOI: IEEEINFOCOM.10.1109/INFOCOM.2006.312.

[31] Saikat Guha and Neil Daswani. *An experimental study of the skype peer-to-peer voip system*. Tech. rep. Cornell University, 2005.

[32] Ryan Paul. *More universities banning Skype*. Sept. 2006. URL: https://arstechnica.com/uncategorized/2006/09/7814 (Accessed 19. Nov. 2018).

[33] *Ban Skype*. Nov. 2005. URL: http://blog.tmcnet.com/blog/rich-tehrani/voip/ban-skype.html (Accessed 19. Nov. 2018).

[34] Zack Whittaker. 'Skype ditched peer-to-peer supernodes for scalability, not surveillance'. In: *ZDNet* (Dec. 2015). URL: https://www.zdnet.com/article/skype-ditched-peer-to-peer-supernodes-for-scalability-not-surveillance (Accessed 4. Dec. 2018).

[35] *Microsoft Investor Relations - Acquisitions History*. URL: https://www.microsoft.com/en-us/Investor/acquisition-history.aspx?CollectionId=null&year=2011 (Accessed 4. Dec. 2018).

[36] Peter Bright. *Skype finalizes its move to the cloud, ignores the elephant in the room*. July 2016. URL: https://arstechnica.com/information-technology/2016/07/skype-finalizes-its-move-to-the-cloud-ignores-the-elephant-in-the-room (Accessed 4. Dec. 2018).

[37] Nadeem Unuth. 'Skype Changes From P2P to Client-Server Model'. In: *Lifewire* (Nov. 2018). URL: https://www.lifewire.com/skype-changes-from-p2p-3426522 (Accessed 4. Dec. 2018).

[38] Dawn Kawamoto. *Microsoft Moving Skype To Azure, Ditching P2P - InformationWeek*. July 2016. URL: https://www.informationweek.com/cloud/infrastructure-as-a-service/microsoft-moving-skype-to-azure-ditching-p2p--/d/d-id/1326345 (Accessed 4. Dec. 2018).

[39] Sergey Tkachenko. *Microsoft Removes P2P Support From Skype*. Feb. 2017. URL: https://winaero.com/blog/microsoft-removes-p2p-support-skype (Accessed 4. Dec. 2018).

[40] Glenn Greenwald et al. 'Microsoft handed the NSA access to encrypted messages'. In: *the Guardian* (July 2013). URL: https://www.theguardian.com/world/2013/jul/11/microsoft-nsa-collaboration-user-data (Accessed 4. Dec. 2018).

[41] Paul Glazowski. 'NBC Direct To Get a Reboot With Help From Pando'. In: *Mashable* (Feb. 2008). URL: https://mashable.com/2008/02/27/nbc-direct-pando/#QYVS1_EAQgqx (Accessed 21. Nov. 2018).

[42] *More Game Companies Select Pando Networks to Optimize Game Downloads*. Sept. 2009. URL: https://www.tmcnet.com/usubmit/2009/09/16/4373266.htm (Accessed 21. Nov. 2018).

[43] *Pando Networks/Riot Games deal*. Feb. 2011. URL: https://www.gamesindustry.biz/articles/pando-networks-riot-games-deal-game-delivery-services-selected-to-bolster-download-performance-of-league-of-legends (Accessed 21. Nov. 2018).

[44] Reuven Cohen. *Content Delivery Cloud (CDC)*. Oct. 2008. URL: http://www.elasticvapor.com/2008/10/cloud-content-delivery-cd.html (Accessed 21. Nov. 2018).

[45] "Not a Cobra". *Pando Media Booster; Uncool.* Feb. 2011. URL: http://forums.na.leagueoflegends.com/board/showthread.php?t=521081 (Accessed 30. Nov. 2018).

[46] *PMB.exe - how to fix PMB.exe errors*. 2012. URL: http://www.process-information.net/us/pmb-exe.htm (Accessed 23. May 2019).

[47] *Pando Software*. June 2012. URL: https://forums.malwarebytes.com/topic/111098-pando-software (Accessed 23. May 2019).

[48] *Infected with Trojan horse Generic, BackDoor - Page 2 - Virus, Trojan, Spyware, and Malware Removal Help*. Mar. 2013. URL: https://www.bleepingcomputer.com/forums/t/490012/infected-with-trojan-horse-generic-backdoor/page-2 (Accessed 23. May 2019).

[49] *Should I remove Pando Media Booster by Pando Networks?* URL: https://www.shouldiremoveit.com/Pando-Media-Booster-6090-program.aspx (Accessed 23. May 2019).

[50] Jay Geater. *What is PMB.exe and How to Fix It? Virus or Safe?* Aug. 2018. URL: https://www.solvusoft.com/en/files/error-virus-removal/exe/windows/pando-networks-inc/pando-media-booster/pmb-exe (Accessed 23. May 2019).

[51] "Cimitiere". *Cannot install LoL due to PMB? - League of Legends Community*. Nov. 2011. URL: http://forums.euw.leagueoflegends.com/board/showthread.php?t=474810 (Accessed 23. May 2019).

[52] *PMB.exe Windows process - What is it?* URL: https://www.file.net/process/pmb.exe.html (Accessed 23. May 2019).

[53] *P4P Working Group (P4PWG) Membership*. Nov. 2014. URL: http://www.dcia.info/activities/p4pwg/membership.html (Accessed 21. Nov. 2018).

[54] Haiyong Xie et al. 'P4P: Provider Portal for Applications'. In: *SIGCOMM Comput. Commun. Rev.* 38.4 (Aug. 2008), pp. 351–362. ISSN: 0146-4833. DOI: 10.1145/1402946.1402999. URL: http://doi.acm.org/10.1145/1402946.1402999.

[55] Nil Sanyas. 'P4P : le TGV du P2P ou simple chimère?' In: *Next INpact* (Mar. 2008). URL: https://www.nextinpact.com/archive/42536-P4P-TGV-P2P-simple-chimere.htm (Accessed 27. Nov. 2018).

[56] *Microsoft Investor Relations - Acquisitions History*. URL: https://www.microsoft.com/en-us/Investor/acquisition-history.aspx?CollectionId=null&year=2013 (Accessed 26. Nov. 2018).

[57] Peter Chapman. *Microsoft Acquires Pando - Next Xbox Uses Rumoured*. Mar. 2013. URL: https://www.thesixthaxis.com/2013/03/14/microsoft-acquires-pando-next-xbox-uses-rumoured (Accessed 26. Nov. 2018).

[58] Tom Warren. 'Microsoft to deliver Windows 10 updates using peer-to-peer technology'. In: *Verge* (Mar. 2015). URL: https://www.theverge.com/2015/3/15/8218215/microsoft-windows-10-updates-p2p (Accessed 26. Nov. 2018).

[59] Saroj Kar. *Spotify abandoning P2P in favor of a more traditional dedicated architecture - SiliconANGLE*. Apr. 2014. URL: https://siliconangle.com/2014/04/22/spotify-abandoning-p2p-in-favor-of-a-more-traditional-dedicated-architecture (Accessed 21. Nov. 2018).

[60] Gunnar Kreitz and Fredrik Niemela. 'Spotify–large scale, low latency, P2P music-on-demand streaming'. In: *Peer-to-Peer Computing (P2P), 2010 IEEE Tenth International Conference on*. IEEE. 2010, pp. 1–10.

[61] Christian Brazil Bautista. 'Spotify to shut down its old P2P network to prevent music from eating your bandwidth'. In: *Digital Trends* (Apr. 2014). URL: https://www.digitaltrends.com/music/spotify-shuts-p2p-network-prevent-music-eating-bandwidth (Accessed 5. Nov. 2018).

[62] *Unadvertised P2P feature*. May 2013. URL: https://community.spotify.com/t5/Desktop-Windows/Unadvertised-P2P-feature/td-p/398612 (Accessed 21. Nov. 2018).

[63] Ricardo Vice Santos. *Spotify: P2P music streaming*. May 2011. URL: https://www.slideshare.net/ricardovice/spotify-p2p-music-streaming (Accessed 27. Nov. 2018).

[64] Ernesto. *Spotify Starts Shutting Down Its Massive P2P Network*. Apr. 2014. URL: https://torrentfreak.com/spotify-starts-shutting-down-its-massive-p2p-network-140416 (Accessed 27. Nov. 2018).

[65] *Spotify now available to everyone in the UK*. Feb. 2009. URL: https://web.archive.org/web/20161108195306/https://news.spotify.com/no/2009/02/10/spotify-now-available-to-everyone-in-the-uk (Accessed 12. Mar. 2019).

[66] *Back to invites for a while in the UK*. Sept. 2009. URL: https://web.archive.org/web/20161108195232/https://news.spotify.com/no/2009/09/10/back-to-invites-for-a-while-in-the-uk (Accessed 12. Mar. 2019).

[67] Romain Dillet. 'Spotify Removes Peer-To-Peer Technology From Its Desktop Client'. In: *TechCrunch* (Apr. 2014). URL: https://techcrunch.com/2014/04/17/spotify-removes-peer-to-peer-technology-from-its-desktop-client/?guccounter=1 (Accessed 27. Nov. 2018).

[68] Scott Carey. *How Spotify migrated everything from on-premise to Google Cloud Platform*. July 2018. URL: https://www.computerworlduk.com/cloud-computing/how-spotify-migrated-everything-from-on-premise-google-cloud-platform-3681529 (Accessed 28. Nov. 2018).

[69] Jordan Novet. 'Spotify said it's relying more on Google's cloud even as the companies compete in music streaming'. In: *CNBC* (Feb. 2018). URL: https://www.cnbc.com/2018/02/28/spotify-is-relying-googles-cloud-according-to-ipo-filing.html (Accessed 28. Nov. 2018).

[70] Matt Weinberger. 'Spotify is making a big switch, and it's a huge win for Google'. In: *Business Insider* (Feb. 2016). URL: https://www.businessinsider.com/spotify-switches-to-google-cloud-platform-2016-2?r=US&IR=T&IR=T (Accessed 28. Nov. 2018).

[71] *BigQuery - Analytics Data Warehouse | BigQuery*. URL: https://cloud.google.com/bigquery (Accessed 28. Nov. 2018).

[72] *Paul Lamere on Twitter*. Feb. 2016. URL: https://twitter.com/plamere/status/702168809445134336 (Accessed 28. Nov. 2018).

[73] *MPJ on Twitter*. Feb. 2018. URL: https://twitter.com/mpjme/status/702167231623516160 (Accessed 28. Nov. 2018).

[74] "sactori". *Spotify makes my games lag*. Sept. 2014. URL: https://community.spotify.com/t5/Desktop-Windows/Spotify-makes-my-games-lag/td-p/699856 (Accessed 21. Nov. 2018).

[75] "JRRR". *Limiting bandwidth*. Apr. 2013. URL: https://community.spotify.com/t5/Accounts/Limiting-bandwidth/td-p/376676 (Accessed 21. Nov. 2018).

[76] "blaszergling". *Music won't play smoothly at some hours.* Sept. 2013. URL: https://community.spotify.com/t5/Desktop-Windows/Music-won-t-play-smoothly-at-some-hours/td-p/543452 (Accessed 21. Nov. 2018).

[77] "effzed". *Spotify keeps stopping - streaming issues - i'm off!!* May 2012. URL: https://community.spotify.com/t5/Desktop-Windows/Spotify-keeps-stopping-streaming-issues-i-m-off/td-p/63517 (Accessed 21. Nov. 2018).

[78] Prithula Dhungel et al. 'Measurement Study of Xunlei: Extended Version'. In: (2011).

[79] Prithula Dhungel et al. 'Xunlei: Peer-Assisted Download Acceleration on a Massive Scale'. In: Mar. 2012. DOI: 10.1007/978-3-642-28537-0_23.

[80] "IPOdesktop". 'IPO Preview: Xunlei Ltd.' In: *Seeking Alpha* (July 2011). URL: https://seekingalpha.com/article/280026-ipo-preview-xunlei-ltd (Accessed 26. Nov. 2018).

[81] Ge Zhang et al. 'Unreeling Xunlei Kankan: Understanding Hybrid CDN-P2P Video-on-Demand Streaming'. In: *IEEE Transactions on Multimedia* 17.2 (Feb. 2015), pp. 229–242. ISSN: 1520-9210. DOI: 10.1109/TMM.2014.2383617.

[82] Joan Calvet. *Win32/KanKan – Chinese drama*. Oct. 2013. URL: https://www.welivesecurity.com/2013/10/11/win32kankan-chinese-drama (Accessed 24. May 2019).

[83] "LoneStar". *Kankan*. URL: https://www.enigmasoftware.com/kankan-removal (Accessed 24. May 2019).

[84] *Xunlei Announces Completion of Strategic Divestment of Xunlei Kankan*. July 2015. URL: https://globenewswire.com/news-release/2015/07/15/752205/10141783/en/Xunlei-Announces-Completion-of-Strategic-Divestment-of-Xunlei-Kankan.html (Accessed 7. Dec. 2018).

[85] *Sliver.tv*. Jan. 2019. URL: https://www.crunchbase.com/organization/sliver-tv#section-overview (Accessed 17. Jan. 2019).

[86] Lucas Matney. 'Sliver.tv is a VR Twitch for your favorite eSports titles'. In: *TechCrunch* (Aug. 2016). URL: https://techcrunch.com/2016/08/24/sliver-tv-is-a-vr-twitch-for-your-favorite-esports-titles (Accessed 17. Jan. 2019).

[87] Theta Labs. 'Blockchain meets esports: over 1 Million Theta tokens earned on SLIVER.tv in 30 days'. In: *Medium* (Feb. 2018). URL: https://medium.com/theta-network/blockchain-meets-esports-over-1-million-theta-tokens-earned-on-sliver-tv-in-30-days-1ab5f6d7af05 (Accessed 17. Jan. 2019).

[88] 'Decentralized video streaming, powered by users and an innovative new blockchain version 1.7'. In: *Theta Network* (May 2018). URL: https://whitepaper.io/document/72/theta-token-whitepaper (Accessed 21. Jan. 2019).

[89] *How WebRTC Is Revolutionizing Telephony*. Feb. 2014. URL: http://blogs.trilogy-lte.com/post/77427158750/how-webrtc-is-revolutionizing-telephony (Accessed 21. Jan. 2019).

[90] *SLIVER.tv on Twitter*. Aug. 2018. URL: https://twitter.com/slivertv360/status/1029444223748075521 (Accessed 24. May 2019).

[91] *Tfuel should be allowed to be withdrawn from sliver.tv following new addition to site.* Apr. 2019. URL: https://www.reddit.com/r/theta_network/comments/ba3o7m/tfuel_should_be_allowed_to_be_withdrawn_from (Accessed 24. May 2019).

[92] *How legit is sliver.tv?* Feb. 2018. URL: https://www.reddit.com/r/GlobalOffensive/comments/80kxvx/how_legit_is_slivertv (Accessed 24. May 2019).

[93] *Cyber-The-Vote-Cost-Benefit-Analysis-1*. Jan. 2016. URL: https://openparachute.wordpress.com/2016/01/28/new-study-finds-community-water-fluoridation-still-cost-effective/cyber-the-vote-cost-benefit-analysis-1 (Accessed 18. Apr. 2019).

[94] André B. Bondi. 'Characteristics of Scalability and Their Impact on Performance'. In: *Proceedings of the 2Nd International Workshop on Software and Performance*. WOSP '00. Ottawa, Ontario, Canada: ACM, 2000, pp. 195–203. ISBN: 1-58113-195-X. DOI: 10.1145/350391.350432. URL: http://doi.acm.org/10.1145/350391.350432.

[95] R. Anandhi and K. Chitra. 'A Challenge in Improving the Consistency of Transactions in Cloud Databases - Scalability'. In: *International Journal of Computer Applications* 52 (Aug. 2012), pp. 12–14. DOI: 10.5120/8172-1485.

[96] Alejandro Reyes. 'Bitcoin Scalability Solutions'. In: *Medium* (Jan. 2018). URL: https://medium.com/@reyesale/bitcoin-scalability-solutions-f5686ffd2ba4 (Accessed 18. Apr. 2019).

[97] Frederick P. Jr. Brooks. 'No Silver Bullet Essence and Accidents of Software Engineering'. In: *Computer* 20.4 (Apr. 1987), pp. 10–19. ISSN: 0018-9162. DOI: 10.1109/MC.1987.1663532.

[98] Axel Grewe et al. 'Automotive Software Product Line Architecture Evolution: Extracting, Designing and Managing Architectural Concepts'. In: Jan. 2017, pp. 203–222.

[99] Lynda Bourne. *Credibility*. Apr. 2013. URL: https://stakeholdermanagement.wordpress.com/2013/04/27/733 (Accessed 6. May 2019).

[100] *How can I control how much data Netflix uses?* Feb. 2019. URL: https://help.netflix.com/en/node/87 (Accessed 12. Feb. 2019).

[101] Mitja Rutnik. *How much data does Spotify use? — probably less than you think*. Oct. 2018. URL: https://www.androidauthority.com/spotify-data-usage-918265 (Accessed 13. Feb. 2019).

[102] Jared Wray. 'Where's The Rub: Cloud Computing's Hidden Costs'. In: *Forbes* (Feb. 2014). URL: https://www.forbes.com/sites/centurylink/2014/02/27/wheres-the-rub-cloud-computings-hidden-costs/#1b7fcb155f00 (Accessed 26. Mar. 2019).

[103] Brent Kelly. *Preparing for Disruption with WebRTC*. May 2013. URL: https://www.nojitter.com/preparing-disruption-webrtc (Accessed 23. Apr. 2019).

[104] Avi Asher-Schapiro. 'YouTube and Facebook Are Removing Evidence of Atrocities, Jeopardizing Cases Against War Criminals'. In: *Intercept* (Nov. 2017). URL: https://theintercept.com/2017/11/02/war-crimes-youtube-facebook-syria-rohingya (Accessed 21. May 2019).

Glossary

Content Delivery Network (CDN) A network of distributed servers that are optimized for content delivery.

Internet service provider (ISP) A provider of services for accessing or participating in the Internet.

Peer-to-peer (P2P) A distributed architecture involving the coordination and cooperation of peers.

Point of Presence (PoP) A server or node that is located geographically close to users.

Video on Demand (VoD) Video content that is delivered on demand.

Virtual reality (VR) A technology presenting a three-dimensional virtual world that can be interacted with, as well as simulating as many senses as possible.

Voice over IP (VoIP) A group of technologies that enable voice communications over the Internet.

Web Real-Time Communication Technology that enables audio and video delivery through P2P in web browsers.

Availability The measure for how much a system is operable at any given time.

BitTorrent An P2P protocol and application used for file sharing.

Blockchain A technology that is used to store distributed data, requiring peers to verify each change or transaction that happens to the data.

Bot Automated programs that perform simple and repetitive tasks.

Caching The act of storing data that is often used in faster storage.

Caching Server/Node A caching server/node that duplicates content it receives, in order to distribute it further.

Client A user of a service. Sometimes used as hardware or software used to connect to a service or solution.

Client-server A distributed architecture consisting of clients connecting to centralized server and requesting its resources.

Cloud computing A service or platform that offers computational resources.

Compression The act of reducing sizes of data, by either elimination redundant data or unnecessary or less important information.

Content Information, data or entertainment that is delivered to a user.

Content delivery The act of transmitting content to users.

Cost-effective Effective or productive in relation to its cost.

Data usage How much data is transmitted from a device, often a cellular device.

Distributing service A service that distributes content.

High production value Used to signify that a project have much funding, a large crew and a lot of work and thought put into it.

Hybrid Architecture A software architecture that consists of a combination of two or more distributed architectures. In this case, a combination of client-server and peer-to-peer architectures.

Last-mile The last stretch of Internet infrastructure before connection to user devices.

Live Content that is being delivered while it is still being produced.

Maintainability The ease with which a product can be maintained.

Media A type of content that consists of video, images, sound or any combination thereof.

Media streaming The act of streaming media such as sound or video over the internet.

Mirror Content The act of duplicating content.

On demand Content that is available in its entirety to be streamed by the user at their own leisure.

Peer A user in a P2P network, which provides its own resources in exchange for utilizing other peers resources.

Peer heterogeneity The disparity between peers in relation to devices, hardware resources and bandwidth speed.

Reliability A measure of how well a user can trust a service to work as it should.

Resolution Denotes the quality of media content.

Server farm A large group of networked computers gathered together, acting as servers for services.

Service A software application that is offered to users or businesses. Commonly offered over the Internet.

Software Architecture The fundamental structures of a software system.

Streaming media A way to transmit media content over the internet, where the content is presented to the receiver while it is still being transmitted.

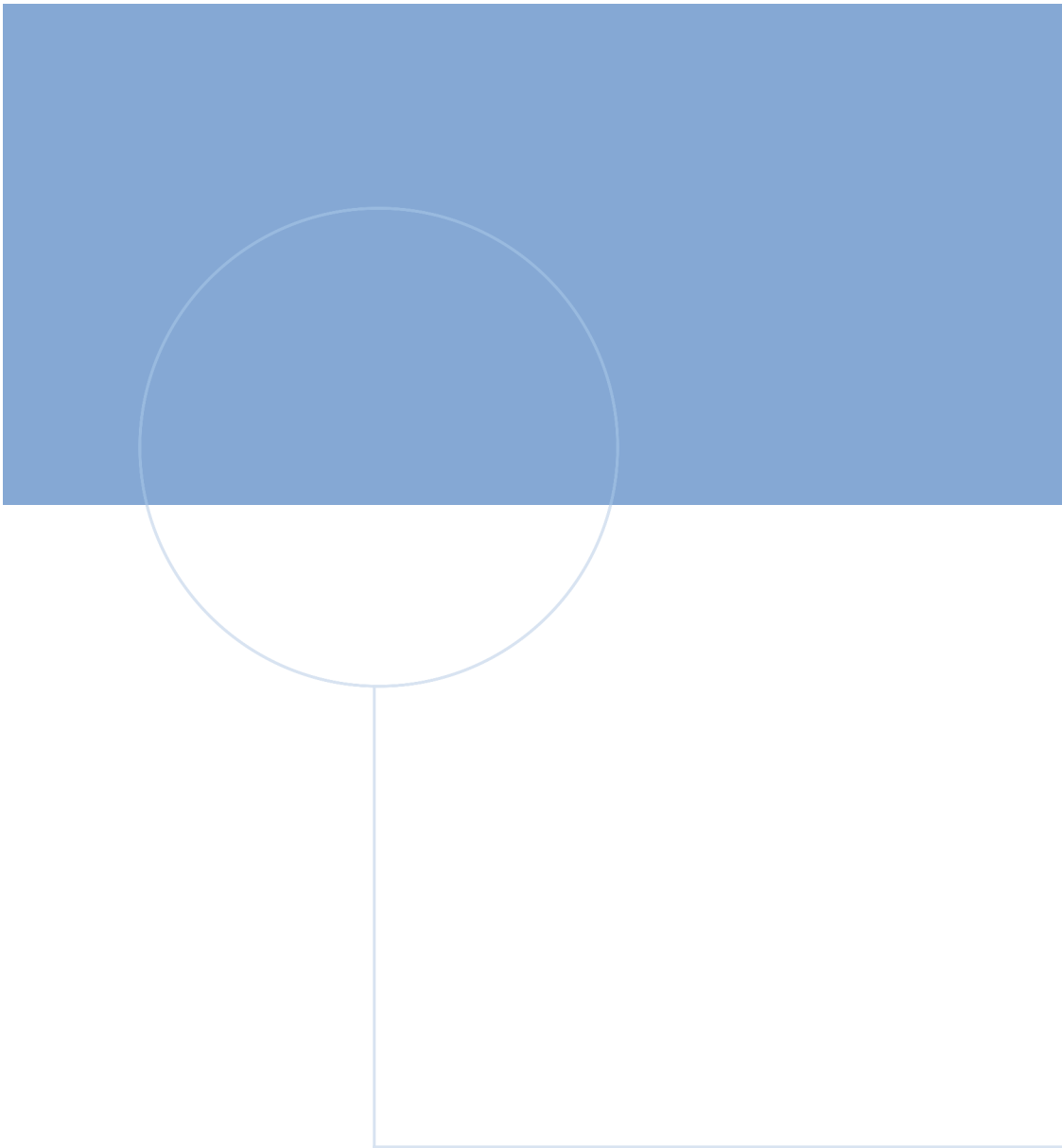
Stutter An anomaly in streaming media causing noticeable disruptions in quality or presentation of content.

User base The people who use a certain product or service.

User-created content Content that is created by users and distributed by a service.

Virtual currency Describes digital money that is issued and commonly controlled by its developers.

Virtualization A technology that abstracts hardware machines and resources and forms them into virtual nodes that can be used for tasks.



NTNU

Kunnskap for en bedre verden