Detection of user patterns in live streaming media during the London Olympic Games

Robin Larsson
ic08rl3@student.lth.se

Department of Electrical and Information Technology
Lund University

Advisor: Maria Kihl, EIT

April 3, 2013

Printed in Sweden
E-huset, Lund, 2013

Abstract

As the Internet continues to grow so does the need for smarter networks that can cater the ever-growing amount of traffic. Streaming media has in just a few years become a major part of the Internet's data flow, but the distribution of streaming media is still done in the same, inefficient way traditional WWW content is distributed. This leads to a potential decrease in both quality of service and quality of experience. In order to make networks smarter, it is important to make them able to understand their users. By dividing users into smaller clusters based on their viewing preferences it will be easier to predict which content will become popular, which in turn can be used for increasing the users' quality of experience e.g., by caching the content in a closely located distribution server.

In this thesis, an analysis of the Swedish state-owned streaming video service *SVT Play* is done based on data traffic collected in a Swedish city during the 2012 London Olympics (2012-07-27 – 2012-08-12). More general analyses to understand user behaviour are done separately for general SVT Play data and data related to the Olympic Games. Methods for clustering users on a content access basis are then suggested.

Keywords: Online user clustering, Content popularity prediction, Streaming media distribution

Sammanfattning

I takt med att användandet av Internet och behovet av höga överföringshastigheter ökar behövs nätverkslösningar som är smartare än de vi har idag. Att bygga ut nätverken med fler optiska fibernät är inte en ekonomisk lösning, särskilt inte när Internet växer så snabbt som det gör idag och inte visar några som helst tecken på att sakna ned.

Ett sätt att minska mängden data som överförs mellan olika nätverk är att använda sig av lokala lagringsservrar som känner av vilket innehåll på Internet som håller på att bli populärt. Om man kan spara ner t.ex. ett *YouTube*-klipp i en stad där många tittar på just det klippet, behöver invånarna i den staden inte skicka sina förfrågningar hela vägen till YouTubes egna servrar. Förutom att ta bort belastning från de stora nätverken som går t.ex. mellan städer innebär denna lösningen dessutom att användaren får hem videon snabbare.

I denna undersökning har en mätning gjorts i ett stadsnät i en medelstor svensk stad. All data som skickats till och från videotjänsten *SVT Play* har sparats ner och analyserats. Mätningarna gjordes under sändningarna från OS i London 2012. SVT hade mycket sändningar som var exklusiva för SVT Play och som inte visades på de vanliga TV-kanalerna.

I analysen visade det sig att användare kan grupperas i olika intressegrupper beroende på vilka videor de tittat på. Genom att veta vilka intresseområden en grupp användare har, kan man också automatiskt få "lagringsstationer" nära just dessa användare så att de kan få snabb tillgång till sådant de med hög sannolikhet kommer titta på. Det visade sig t.ex. att en stor grupp användare endast tittar på barnprogram och ingenting annat.

Det finns redan idag metoder för att spara program på lokala servrar men de flesta har bristen att de inte vet vad som kommer bli populärt förrän det redan har blivit populärt. Detta är ett problem som skulle kunna lösas åtminstone delvis med hjälp av användargrupperingen som visas i denna analys.

Acknowledgements

To those who have helped or supported me while writing this thesis:

Maria Kihl, for being a good and encouraging supervisor with many ideas and pointers,

Andreas Aurelius, for supporting me through everything and giving invaluable feedback, for making starting up a painless process and for the cake,

Jolina Haberkamm, for being my travelling companion when running away to the other side of the world and also being my "thesis buddy",

Niclas Unnervik, for the same reasons and for the many unit discussions,

Manxing Du, for the help, for the introduction and for making me feel a little less lost,

everyone else at Acreo who was involved in the project, showed interest, had questions, comments or ideas,

my parents, for putting up with me all these years,

谢谢小微这几年给我的鼓励、支持和动力。

My friends who have shown interest,

anyone else I have forgotten.

Thank you, everyone!

Robin Larsson
Lund, 2013-02-12

Table of Contents

List of Figures

List of Tables

Preface

This master's thesis is typeset using LaTeX 2_ε. Its number and unit formatting is done according to the scientific standard for quantities and measurement described in [1]. Formatting of dates and times is done according to [2] and are, unless otherwise noted, in Central European Summertime (UTC+2:00, CET+1:00). References and citations follow the formatting suggested by IEEE as described in [3] as closely as possible. The basic unit for HTTP communication is called a *message* and its contents *entities*, as per the HTTP/1.1 RFC, [4].

Chapter 1
Introduction

The Internet is a big place. It is also a very crowded place. New technology means faster bit transfer rates and faster bit transfer rates lead to more new technology. In the end of the previous century, around 1990 when the Internet started its real growth, who could have imagined that they would be watching streaming video over the Internet just a few years after they had had to wait for two hours to download a short sound clip with their 56.6 kB/s modem? Indeed, the Internet has come a long way since then and streaming video is becoming one of the major contributors of Internet traffic in networks. Some events are bound to be more popular than others, in streaming media just like on the old-fashioned television. One example of an event that surely is popular with almost everyone is the *Olympic Games*. In 2012, the *Games of the XXX Olympiad* took place in London and for the first time ever in Olympic history, the games were *extensively* covered in streaming video services worldwide.

1.1 Internet usage

The Internet has been growing for many years now. In 2011, the total *Internet Protocol* (IP) traffic per month was 30.734 EB (one exabyte is 10^{18} bytes). Of this, 597 PB (1.94%) was mobile traffic (one petabyte is 10^{15} bytes). This is expected to increase to 110.282 EB by 2016, of which 10.804 EB (9.80%) will be mobile traffic [5] and by 2018 around 13 EB will be transfered by mobile units [6]. New technology which leads to higher bitrates over both mobile networks and fixed networks has made it not only possible, but also very plausible to stream video and other content. Against this background and with this new technology, it is interesting to see how popular it has been to watch the Olympic Games over the Internet. Earlier years, it has been difficult to stream video content to mobile phones because of the low bitrates in the mobile networks. Now it is possible to do so with fair quality. Since people can not always be at their homes watching the games on their television sets, maybe they will watch it on-the-go on their mobile phones, instead.

The last few years, streaming media from the Internet has become more and more popular with sites like *YouTube* that was released in 2005. Several TV stations have gone with the times and now allow streaming of content at any time and on a multitude of platforms, such as phones, IPTVs, tablets, *etc.* This

content has earlier been accessible only in real-time via television receivers. Some VoD (video on demand) suppliers, such as SVT Play, even offer content that is not televised and only accessible via the online streaming service.

The way streaming services work today are highly inefficient. Each end user needs their own unicast connection to the server and this can put a huge load on the transmission channels. This will first and foremost be a problem in the mobile networks, that are already struggling with data traffic overload since the smartphones entered the market [7]. One way to partially remedy the problems is with smarter networks. If a lot of users in a certain area like watching the same content on VoD services, it could be possible to cache it close to the users and send it to them when they request it. In some cases it could even be possible to predict what the users will be watching, and pre-cache content before any user even requests it. This is especially true for live streams, that by their nature have to be watched in real-time. If one could predict the popularity of streams in certain areas or for certain demographics, one could do away with the necessity for a unicast connection to every user.

1.2 SVT Play

SVT Play (`www.svtplay.se`) is an online streaming service owned by *Sveriges Television* (SVT), a Swedish national television broadcaster. It is free for anyone to use but access to some of the content is limited to users within Sweden. SVT is funded partially by a fee that is compulsory for everyone with a TV receiver to pay. This removes the need for advertising and this ensures SVT can be idependent from commercial companies, political parties and other interest groups and therefore remain unbiased. Copyright laws may, however, stop SVT Play from being able to stream all shows to all platforms or all countries [8].

This was the case for the Olympic Games. In the U.S., NBC paid $1 180 000 000 for the sole rights to broadcast the Olympic Games [9]. SVT Play had to block viewers from outside of Sweden. During the period of the London Olympic Games, SVT Play had several live streams available for viewing. Some of the programmes shown on live streams were SVT Play exclusive; they were not broadcast on regular TV.

1.3 IPNQSIS

IPNQSIS, or *IP Network Monitoring for Quality of Service Intelligent Support*, is a project whose main objective is measuring the *quality of experience* for end users. IPNQSIS in turn is part of the Celtic-Plus initiative. Celtic-Plus "is an industry-driven European research initiative to define, perform and finance through public and private funding common research projects in the area of telecommunications, new media, future Internet, and applications & services focusing on a new 'Smart Connected World' paradigm" [10].

IPNQSIS is unique in that it takes both the network performance, *quality of service* (QoS) and the end user's experience, *quality of experience* (QoE) into consideration when trying to measure the quality of communication systems. The

project is a continuation on earlier projects such as *Traffic Measurements and Models in Multi-Service Networks* (TRAMMS), which did not focus as much on user patterns and user behaviour.

Chapter 2
Background

This chapter provides the motivation behind this thesis, as well as the background required to understand it.

2.1 Motivation

When the Internet first started evolving around the year 1969, it was a small network between a few universities in the U.S. and was never meant to be used by almost everyone the way it is today (something that is made painfully evident by IPv4 addresses running out). With the advent of the *killer applications* of the Internet, the *World Wide Web* and *e-mail* for example, the user base began growing incredibly fast. Figure 2.1 shows the amount of users connected to the Internet during 2001–2011.

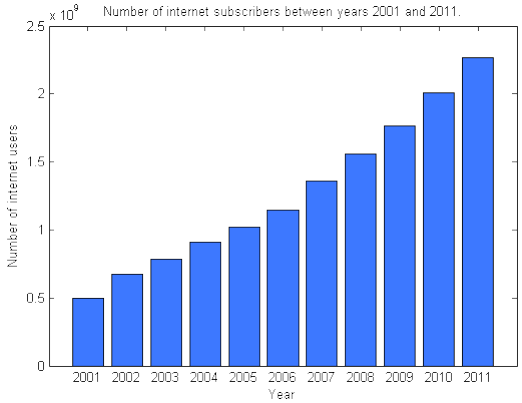

Figure 2.1: In the last few years the number of users on the Internet and with it the amount of data transfered, has increased manyfold [11].

Accessing Internet with mobile phones is a new and ever-growing paradigm. According to [6] the number of subscriptions for *Long Term Evolution* (LTE) or *High Speed Packet Access* (HSPA) technologies have increased greatly since 2009,

while the number of subscribers for the *Global System for Mobile Communications* (GSM) and *Enhanced Data rates for GSM Evolution* (EDGE) services will start to decline around 2013. By 2018, a great majority of all subscribers will use one of the technologies with a fast data transfer rate. Between 2012 and 2018 it is believed the amount of mobile network data from tablets will increase by a factor 40, smartphones by a factor 14 and personal computers by a factor of 7, giving a total of a mobile data growth by a factor 12. The huge growth is driven mainly by video streaming over mobile Internet. The prognosis from [6] can be seen in Figure 2.2. A prognosis for the number of subscriptions in the world can be seen in Figure 2.3. Figure 2.4 shows some historical data for the amount of data transfered in mobile networks, both voice and Internet.

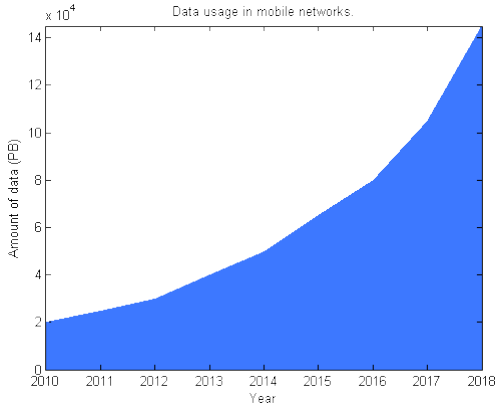

Figure 2.2: The predicted growth of the data usage in mobile networks between years 2010 and 2018 [6].

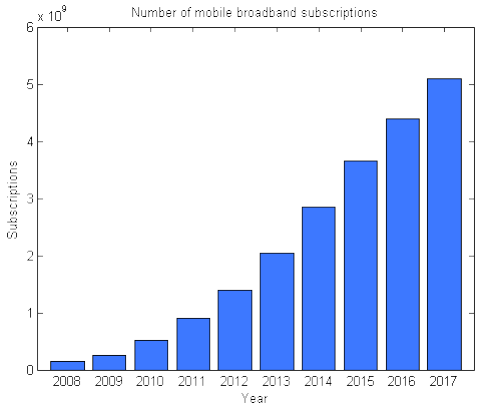

Figure 2.3: The predicted growth of the number of mobile network subscribers between years 2010 and 2017 [6].

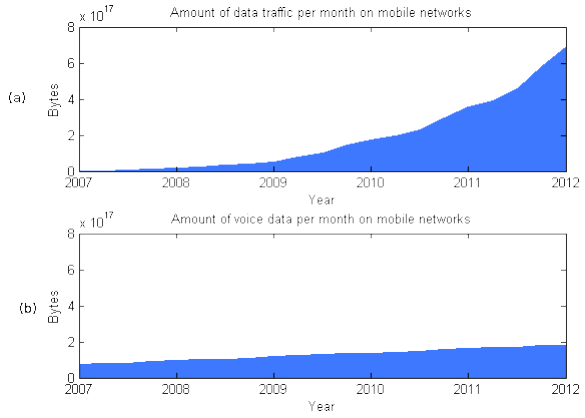

Figure 2.4: (a) The amount of Internet data traffic per month in mobile networks from 2007–2012.
(b) The amount of voice data traffic per month in mobile networks [6].

The technology to do something about this growing problem already exists and it is in the Internet service providers' (ISPs) interests to decrease the amount of data traffic. If networks become smarter, there might not be any need to redimension the existing networks to handle increasing amounts of data, something that will decrease the cost of operation for ISPs. In Figure 2.5 a typical network as they work today can be seen. In (a), three users that are positioned behind the same network node (router, gateway, etc) request the same video, *Video "A"*. The network node is stupid and does not realise they want to see the exact same thing, it is only there to pass the requests on to the next node in the network, until the content server is reached on the other side of the Internet. In (b), the server receives the three requests and treats them as three separate requests, responding to each one of them by starting the stream of Video "A". The obvious flaw in this behaviour is, of course, that three identical copies of the same stream is sent through the entire Internet from the originating server or another server with the requested data, just to eventually end up at more-or-less the same place.

If the network node close to the three users *understood* what it was doing better, it could have a solution for example like the one in Figure 2.6. In that example, the node realises the three users want the same thing, thus sending only one request to the originating content server. The originating content server only responds with one stream. The network node near the users will then distribute it to the users that want to watch it. The solution could be made more general, e.g., all three users might not necessarily need to be connected to the very same router. The network node that passes the request on to the content server could be higher up in the hierarchy. Nodes higher up in the hierarchy could be smarter in the same way. If two network nodes send the same request the next node in line could send just one request to the originating server, and so on.

While the probability of two people requesting the same "chunk" of data at

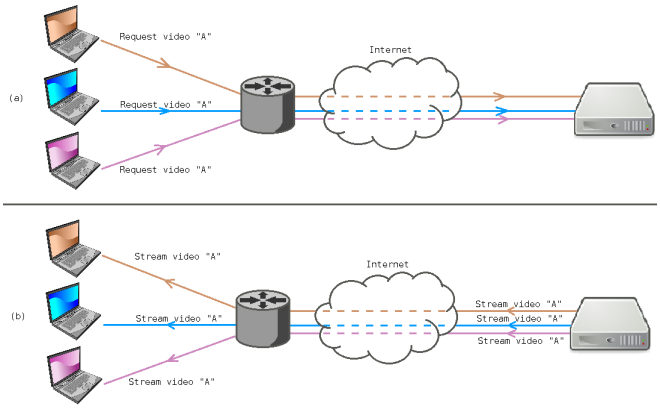

Figure 2.5: Example of a network that is not smart. If three users as pictured are connected to the same gateway and request the same stream (a), the server will send three identical data streams to the terminals (b).

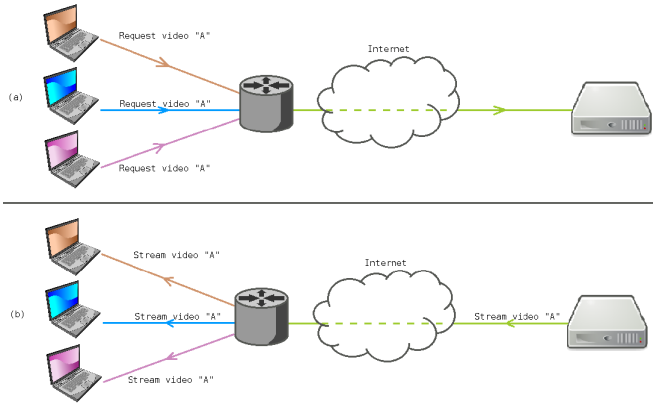

Figure 2.6: Example of a smarter network. (a) Each user requests the same programme. The network node that connects the three terminals is smart and therefore only sends one request to the server. (b) The server responds with only one video stream since only one was requested and the network node will receive and redistribute it to all the viewers.

the same time is rather high if the content is a live stream the same may not be true if the data is VoD. This could be solved by saving copies of content in some local servers. If one user requests a video at one point in time the request will be sent from the user to the server. When the server responds with a stream the nodes on the way back to the user can save that stream to some sort of memory. The next time a request for the same content arrives at the network node with the locally cached copy of the stream the request does not have to travel all the way to the content server. The data can be sent back to the user directly from the network node, thus eliminating the need for the stream to travel between the content server and the network node with the local copy. See Figure 2.7 for an illustration of this idea.

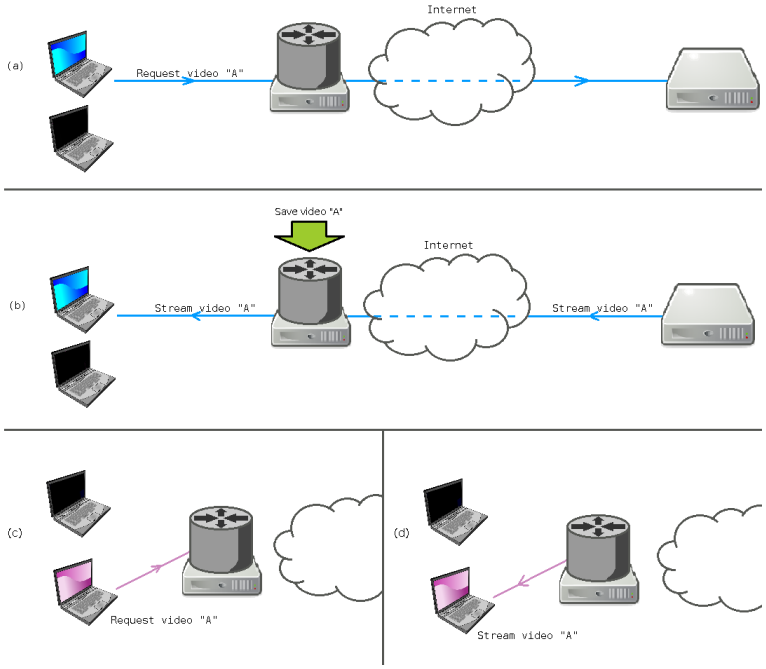

Figure 2.7: (a) A user requests a streaming video. The request goes to the originating server on the other side of the Internet. (b) The originating server starts streaming the content to the user, but a network node close to the user has a buffer to which it can save content. While streaming to the user, it also saves the stream. (c) Another user, close to the first one, requests the same video. (d) Since the node already has the video cached, it starts streaming back directly to the user. No traffic is sent from the originating server.

So-called *Content Delivery Networks* (CDNs) are a partial solution to the problems presented. The original content service providers can utilise the services of second-party companies to distribute media from their servers instead, thus reducing the traffic to and from their own servers significantly. The original content

provider will still have to distribute the content to the CDN servers though. SVT Play, like many other European broadcasters, used the company Qbrick as their CDN of choice [12] during the measurements. However, since then they have changed to another CDN provider, Akamai[13]. When a user requested a stream from SVT Play, they would be "redirected" to a Qbrick server containing the actual video stream. While reducing the traffic to and from the original server, preventing it from being a bottleneck, it is still not certain the CDN servers are close. There are both good and bad things about the use of CDNs, good things include the fact that many servers have the same content, in case the route to one of the servers is cut off there is still a possibility a user can access another server with the same content. Negative things include the fact that it will cost money for the service providers. In appendix A, an example of available streams for a certain piece of content is presented. In the *Universal Resource Indicators* (URIs), both `qbrick` and `cdn` appear. Read more about the same example in section 2.3.3.

2.2 About HTTP

Hypertext Transfer Protocol (HTTP) is the application protocol on which the WWW was built. HTTP/1.1 was described in RFC 2616 [4]. HTTP defines how data is transfered over the Internet and can be used for e.g., images and text. HTTP is an application protocol which means it does *not* describe how the data will reach its destination on the internet, how it will be represented on wire or which application the data stream belongs to. This is handled by layers below the application layer by other protocols than HTTP, such as the *Transmission Control Protocol* (TCP). There are two types of HTTP messages, *requests* and *responses*. Traditionally, one session of HTTP communication is one request and one response. For each "item" on a website, one request–response pair must be transmitted, however in HTTP/1.1, which has been used for many years now, a connection can be kept alive for several requests and responses, thus avoiding setting up a *slow-start* TCP connection between the host and the server for every image, text entity and other items that may appear on a website. An illustration of a request–response pair is shown in Figure 2.8.

A typical request can be seen in Figure 2.9. The request message always starts with a *method*, the most common ones being `POST`, `GET` and `HEAD`. Simplified, one could say that `POST` is for requesting a website after a form or such has been filled in, sending the "post data" to the server. `GET` is similar to `POST`, except it does not have any post data. User arguments can still be sent to the server, but then as a part of the request URL. It is a so-called "safe" method – the request will not change any data on the server and sending the same request several times will yield the same results every time. `HEAD` is similar to `GET`, with the difference that no entity body will be sent with the response. This is useful for obtaining information about a website without having to actually download all of it. The request line also contains the name of the resource that is requested. The HTTP/1.1 string is the version of HTTP used.

The *host line* is required in HTTP/1.1, it tells the server which host the request is intended for. This is required because one IP address may contain a lot

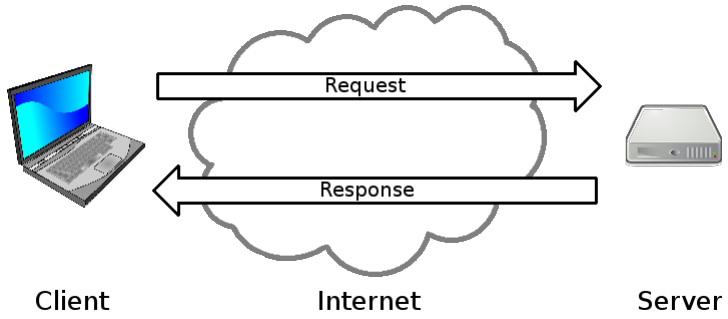

Figure 2.8: A request is a HTTP message sent from a client to the server. The server will then usually send a response containing the data that was requested back to the client. The response might instead contain error messages or other information.

of different servers and a request for e.g., / is ambiguous, every server on a single IP address has a root directory. The rest of the *header lines* can contain a lot of different things that the browser thinks the server will need, for example the `User-Agent` string that tells it which operating system and browser the client is using (which can help the server deliver mobile phone versions of certain websites). They can also tell if the browser supports compression, which character encodings it prefers, etc.

Each of the lines in the header must be terminated by a carriage return and a line feed, \r\n. The header is always terminated by an empty line, a line that *only* contains \r\n. After the header ends, there may be an entity body, depending on the method used.

```
Request line    GET / HTTP/1.1\r\n
   Host line    Host: www.google.com\r\n
                Connection: keep-alive\r\n
                Accept: text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8\r\n
                User-Agent: Mozilla/5.0 (Windows NT 6.1) AppleWebKit/537.17
Header lines       (KHTML, like Gecko) Chrome/24.0.1312.57 Safari/537.17\r\n
                Accept-Encoding: gzip,deflate,sdch\r\n
                Accept-Language: en-US,en;q=0.8\r\n
                Accept-Charset: ISO-8859-1,utf-8;q=0.7,*;q=0.3\r\n
  Empty line    \r\n
```

Figure 2.9: A typical HTTP request message. The request line contains the method (in this case, `GET`), the directory (in this case, root or /) and the HTTP version. The `Host` line is a part of the header that is mandatory in HTTP 1.1. The `User-Agent` string contains information about the operating system and browser used by the client.

An HTTP response is different from a request in a few different ways. Figure 2.10 shows one example of a HTTP response. It is not the response generated by the request in Figure 2.9. There is no method line, it is replaced by the status line. The status line contains the version of HTTP used, followed by a code, for

example 200 which means that everything went OK or the well-known 404, that means the file was not found. A text description of the status message, such as *OK* or *Not found* is also included after the code. Just like the request the response can contain an arbitrary number of header lines if the server wants to tell the client anything, such as when the response was generated or, if an entity body exists, the length of it (`Content-Length`). As with requests, each line must be ended by \r\n and an empty line containing only \r\n ends the header and starts the body. The body can contain pieces of a picture, Hypertext Markup Language (HTML) or basically any other type of data. Big files are split into several HTTP messages, since an ethernet frame has a maximum size and a HTTP message must be encapsuled in one of those.

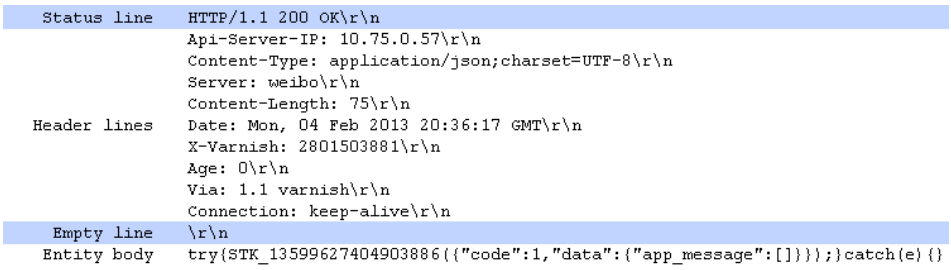

Figure 2.10: A typical HTTP response message. The status line contains a code such as 200 OK or the well-known 404 Not found. An arbitrary amount of header lines follows. An empty line separates the header from the body (if any). Each line is terminated by a *carriage return* and a *line feed*.

2.3 About SVT Play

This section explains some of the technology behind SVT Play. A basic knowledge of things such as how the HTTP requests and responses from the SVT Play server look is required to better understand some of the methodology.

The underlying structure of the online streaming service is subject to constant change. This for many reasons such as, to allow for a better, more efficient streaming service or to disallow downloading of the streaming content to a disc drive (which counts as piracy and is against the law). At the time of the Olympic Games, SVT Play used a technique known as Flash Dynamic Streaming (FDS) for computers or other devices with Flash available. With FDS the video stream is sent over a TCP connection using the Real-Time Messaging Protocol (RTMP). Apple devices do not support FDS and therefore used a completely different technique, HTTP Live Streaming (HLS), which is sent over the standard HTTP protocol. HLS has a few limitations that are not present in FDS, such as bad encryption which in turn means the copyright holders of certain programmes do not allow said programmes to be streamed with the technique [14]. Some other lesser protocols were also available such as Flash Progressive Download, which basically streams a .FLV file in the browser's flash player, and was used for playing short clips.

Since the Olympic Games were broadcast the streaming technologies SVT Play uses have been changed radically.

2.3.1 Message structure

A user connects to the SVT Play server via one of several different types of devices that have different types of operating systems (Android, Windows, Mac OS X, Linux, *etc*). Most users will send statistics information to the server with regular intervals, how these look or if they exist at all varies between different clients and different devices. These messages are used to keep track of viewer statistics on the server-side and can contain a lot of different information.

The server will send lots of messages to the client, most of them are not interesting to look at as they contain things such as HTML data (the code used to render the website), GIF images and other non-stream-related items. The payload content from streams is not interesting in itself, though the length of it can be used to measure the amount of data transfered if one so desires. Messages that are however interesting, are any and all messages with JSON entity bodies. The JSON files contain huge amounts of information such as category, programme name, episode name, broadcast time, *etc*, that can be directly extracted.

2.3.2 Requests

Requests are the messages sent from the user client to the SVT Play server. They are sent as a `GET` message. Depending on the platform used and what type of message is being sent to the server, the user parameters may contain information such as `platform=mobile` and `vformat=m3u8`, which will be used to tell the server which type of stream the client wants to request, since not all players support the same formats. A list of which video formats are available is first requested before the client asks for a stream. Several different types of streams of the same content exist for different framerates, resolutions and video formats for the streaming service to be able to accomodate all types of devices.

2.3.3 Responses

Lots of data is sent from the server-side to the client-side and most of the data is not interesting in this analysis (since most of the messages only contain image data, text data, stream data *etc*). Each and every message with the entity header field `Content-Type` value `application/json` is extremely interesting to look at. SVT Play sends information about a requested video in a *JavaScript Object Notation* (JSON) file structure and most important information for the traffic analysis is contained within.

A JSON file is basically a text file, initiated by the character `{` and terminated by a `}`, containing `key:value` pairs, a so-called *dictionary*. The key is always a string, that is, an entry inside double quotes. A value can however be one of many different data types: *number*, *string*, *boolean*, *array* or another dictionary, which in turn contains more such pairs. These JSON-files are easily parsed using, for example, the `Python` module `json`. An example of what a JSON file looks like can

be seen in Figure 2.11 and an actual example from one of the entity bodies can be found in appendix A.

Each time a user loads a video page and clicks the *Play* button, exactly one of the aforementioned JSON messages is generated by the server. Pausing and unpausing will not generate any additional JSON entities. The JSON file contains, as mentioned above, a "menu" of different streams and the client picks whichever one suits its needs the best. The client will then send another request to the server, which will start the stream.

```
{
   "kings": {
      "Thailand": "Bhumibol Adulyadej",
      "Norway": "Harald V"
   },
   "queens": {
      "Netherlands": "Beatrix",
      "Commonwealth": "Elizabeth II"
   }
}
```

Figure 2.11: An example JSON file. A JSON file is a text file with key and value pairs, or a *dictionary*. The key is a string and the value can be a number, a boolean variable, a string, a list or another dictionary. In this example, the JSON file contains two entries, *kings* and *queens*. The value for each of them is another dictionary containing two more key–value pairs but with string values.

The possibility that someone who has a bad connection and keeps reloading the site and clicks play over and over again exists. This will load several JSON messages containing the same content, but the user has not viewed the programme more than once. There might also be different circumstances that generates more than one JSON entity message. The video player gets access to multiple different streams of the same programme but in different qualities and for different protocols. If the user feels that the stream is lagging, he or she can lower the quality on the video player and the player will instead start receiving one of the lower quality streams. Algorithms can also make calculations automatically to decide which quality is the best for the user's connection [15].

Chapter 3
Previous work

This chapter gives a short introduction of work related to *smarter networks* and online user behaviour from previous and current projects such as TRAMMS and IPNQSIS.

In [16] it is investigated what consumers use their Internet connection for. The target network for traffic measurements was the same network that is analysed in this thesis. The volume of data transfered over broadband connections and which applications were responsible for the generated data were the main *foci*. In the paper it is revealed that 58% of all the inbound data traffic is generated by file sharing applications, such as *BitTorrent*. On the second place with 16% of all the inbound traffic is streaming media.

[17] further examines the user behaviour with regards to popular applications based on the customer's Internet access method: *Fiber to the home* (FTTH), *Cable modem termination system* (CMTS), *Gateway GPRS support node* (GGSN, a mobile technology) and *Digital subscriber line* (DSL). It also measures another important parameter related to QoS and QoE: Roundtrip times (RTT). The roundtrip time is the time it takes for a request to reach the destination and receive a response. Several different networks in Sweden and Spain are included in the measurements and analyses, with most of the users being in Spain. In the Spanish networks, the destination of outbound traffic is in Spain 55% of the times, while the source of inbound traffic is mainly from the U.S. (33%) and Spain (28%). The response times are, not surprisingly, heavily dependant on the physical distance between the source and the destination. The highest RTTs can be found between Spain and Asian or South American countries with times around 0.4 seconds.

With results from the TRAMMS project, the IPNQSIS paper [18] discusses the benefits of local caching of content in CDNs or similar smart network structures. It is explained that CDNs not only reduce the demand for transmission by eliminating the need for each user to have a logical connection all the way to the same content server, but it also increases the availability of content by making it available from multiple different locations. If one of the servers goes down, there will be other servers with local copies of the same data that can take over. It also argues that CDNs increase the performance by decreasing the risk of delay and packet losses.

Furthermore, the term QoE is explained: QoE is a complement to QoS. QoS measures parameters such as packet loss and response times while not focusing at all on the *perceived* quality by the end user. QoE is the quality of a service as perceived, subjectively, by the user. While there is a strong correlation between

QoS and QoE, it is not necessarily the same thing.

The five-year evolution of broadband between the years 2007—2011, again as measured in a Swedish municipality, is analysed in [19]. It shows the evolution of how broadband end-users have been using the Internet over the last few years. The streaming media traffic per end user increased by over 500% between the end of 2008 and mid-2011, compared to the daily web browsing increasing by only 300%. In 2007, 2% of the total network traffic was streaming media, a number that had increased to 13% in the first half of 2011.

[20] measures the *cacheability* of YouTube by analysing the amount of traffic to the video streaming site. It makes contributions to the behaviour of users regarding how and when they watch videos, as well as potential caching gains using two different types of caching: *proxy caching*, in which a server caches content and distributes it to any terminal that requests it, and *terminal caching*, in which the caching is done locally on the computer the first time a user requests a video. As it turns out, users like to rewatch the same videos multiple times, often waiting around 24 hours between each time a video is rewatched.

Some work on user clustering similar to the one in this thesis is done in another master's thesis; [21]. The results are from an analysis of *Facebook* data and in the thesis attempts are made to connect different users to each other by seeing how much they have in common. Those with a lot in common are placed in user clusters, similar to how it is done in this thesis.

While not part of the aforementioned projects, many papers on *recommender systems* in a content distribution context exist, e.g., [22]. When using CDNs in conjugation with automated recommender systems, it was found that the cache hit rate is increased, with the additional gain in using this method decreasing as the cache size increases. Recommender systems based on more or less advanced algorithms exist in many applications and is an important feature of many online stores, such as *Amazon.com*, to give a user browsing the website the same feeling one could get when browsing a real, physical store and find items that the customer did not know that they wanted.

Chapter 4
Methodology

This chapter explains the methods and tools that have been used for measurements and analysis of the data and gives a description of the target network and how the integrity of the users in the target network is protected.

4.1 Data collection

4.1.1 Measured network

All measurements were done in a municipal IP access network in a Swedish town. The users are connected to the Internet via FTTH. Different ISPs operate in the network, giving all the users the possibility to use the ISP they prefer. Each service provider is connected to the network at the Internet edge, but the measurements are done inside the municipal network, meaning all data will first have to pass through the measurement equipment which in turn means that there is no ISP bias in the measurements. This is however rather insignificant in this application; SVT Play is not a service that is intimately connected to any specific ISP. A conceptual sketch of the topology of the network can be seen in Figure 4.1. The measurement tool is connected to the network by the Internet edge with an optical splitter, that will send identical copies of the data to the measurement tool and to the "Internet" beyond the Internet edge. The measured network has a bit more than 5000 homes, all connected with FTTH, and a wide range of users present. The measurements are from a medium-sized city in Sweden. It can be seen as a representative demographic *for Sweden*.

4.1.2 Tools

The tool responsible for collecting all the raw data is called *PacketLogic*. It is a proprietary tool from Procera Networks that analyses data up to the application layer in real-time, so-called Deep Packet Inspection, using its own Datastream Recognition Definition Language (DRDL). The way it has been used in this particular project is similar to how the open-source software `WireShark` works. After telling the tool to start capturing data, it will record each ethernet frame that passes through it. One of PacketLogic's specialities is the large list of applications it "understands" thanks to the DRDL. DRDL provides visibility of applications on layer seven in the *Open Systems Interconnection* (OSI) model. While other tools

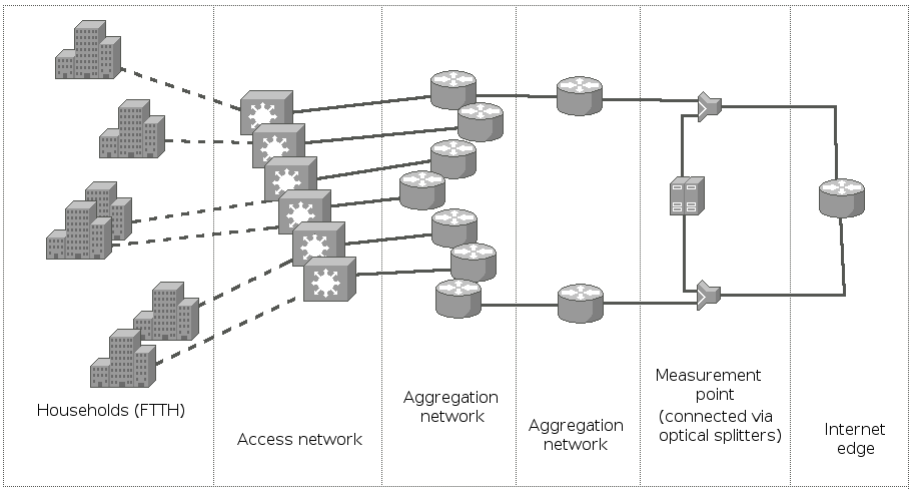

Figure 4.1: Conceptual FTTH network topology [16]. The dashed lines are 100 Mb/s connections, while the solid lines are 1 Gb/s connections. Service providers are connected to the network at the Internet node.

might be able to see what type of application is being used, PacketLogic can often see which specific application it is. Mobile telephone operators and other ISPs can use PacketLogic in their networks for this, for example if a customer has a subscription that includes a certain VoD service the user should not be charged additional data fees for watching it and PacketLogic can keep track of how much data is for the included service in real-time.

While the measurements are running, the PacketLogic will save copies of each frame that passes through it. The frames are all saved to so-called `pcap` files (*packet capture* files). For this measurement, only IP traffic has been recoded. More specifically only IP traffic to and from SVT Play will show up in the pcap files.

4.1.3 Anonymisation

After the raw data has been collected by PacketLogic and saved as pcap files, Swedish personal integrity laws require the anonymisation of personal data such as Internet traffic before it can be analysed further. Before leaving the servers on which the measured data is contained, all MAC addresses and IP addresses, both client-side and server-side, have been *hashed*. A hash function takes a known sequence of bits and transforms them into something that looks completely different from the original bit stream. The hash function used is a 128-bit function, meaning that an input string, regardless of length, returns a 128-bit value (32 hexadecimal digits). See Figure 4.2. A good hash function has a few main properties: Calculating the hash from a given input is fast, it is *very difficult* to find the message of a given hash and it is *very difficult* to find two messages with the same

hash. Unfortunately, there is no way to change the *very difficult*s to *impossible*s and while new hash functions are invented all the time, new ways to crack them also emerge.

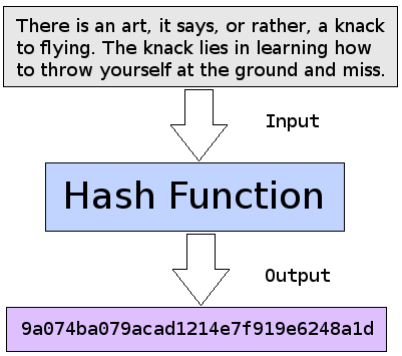

Figure 4.2: A hash function takes a file, a text stream or another binary representation of anything. A good hash algorithm makes the original input *very difficult* to find out. Shown here is an MD5 hash function hashing a quote from the *Hitchhiker's Guide to the Galaxy*, giving a 128-byte binary string as output. Using MD5 for passwords and other sensitive data is not adviced; it has been severely compromised.

The hash function used for MAC addresses and IP addresses is of course good enough to completely hide a user. The hashed values are also all different from each other, meaning two users can be told apart, even if we do not know anything about which two users they are. Henceforth, the terms *IP address* and *MAC address* will be used to mean *hashed IP address* and *hashed MAC address*, respectively.

4.1.4 Data filtering

Copious amounts of data is generated and since the only data of interest in this thesis is the data to and from SVT Play, all other data has been filtered out. All the captured information has then been translated from pcap to text and then been transfered to JSON files. Three different kinds of JSON files were generated by the program that did the filtering; `request`, `response` and `payload`. The request and response files are basically identical and they contain the *header* information from all data traffic to and from the SVT Play servers. For requests the source host will be the client, the person watching a stream. For responses the source host will instead be the SVT Play server. In most other respects, messages from each side are essentially the same, except what differs in the specification of a request and a response, as described earlier.

Some of the responses from the server come with JSON entity bodies. If they do, the entire entity body is saved into a payload file. This is the only time a payload passes through the filter, payloads of all other messages have been ignored. The files that contain the responses from the servers will, if the user requested to

start a video, send a response with information about that video. If this happens, that means someone started a video, and the response file will contain a reference to another file that contains the generated payload file for that particular response, making a response message and a JSON payload easy to match.

There is also a fourth category of files, but they are not in the JSON format. PacketLogic generates *connection logs* while measuring traffic. Each line in a connection log file is one connection. Each connection has two unix timestamps; one for the connection start time and one for the connection termination time. It also contains the IPs and ports of both source and destination, but no MACs.

4.2 Data analysis

4.2.1 Tools for analysis

I have selected `Python` as the language to use for parsing data in this project. With the relatively small amount of data involved the lower performance of Python in comparison to lower-level languages is not an issue. Furthermore, while the execution time of a script in Python is longer than the corresponding program in e.g., `C++`, this is *by far* made up for in time saved writing the scripts.

Plots and graphs, correlations and some other calculations are done in the wonderful tool that is `MATLAB`. Also used for correlation coefficients and p-values is `SciPy`, a library of mathematical tools for Python.

4.2.2 Definitions

For most measurements, a user has been defined as a MAC address. Most users will be behind a router where there are several different devices that access the Internet even if they show up as having the same MAC address. This is not a one-to-one representation of the truth. One user can of course have several different devices (phone, computer, tablet) or dual-boot, but then again it is also possible that two users share the same device. All-in-all, a lot of MAC addresses contain one computer and one phone and there is not a whole lot of users that have a great multitude of different operating systems and terminals. Furthermore, it is highly possible that several users are using the same terminal at the same time or at different times. Therefore, it will be assumed that each unique MAC is one unique user.

There are other ways one could define a user, such as a tuple of the MAC address and a unique `User-Agent` string in the HTTP request headers from the client. This definition leads to other problems: Two browsers on the same computer will count as two different users and a lot of people use both their phone and their computer, meaning an over-representation of the number of users. It has therefore been decided for this application that one MAC address is *approximately* the same as one user.

In the connection logs it is unfortunately not possible to see the MAC addresses of the users, nor the user agent strings, but the IP address is visible. This means that one user has been defined as one IP in that case. This is a bit dangerous and

of course not true at all, but it should still give a good enough picture of during which periods streaming has been more or less popular.

There are two categories of measurements done on the data: *General* and *Olympics specific*. The former case means that all data from SVT Play, including non-sports categories, has been used for the measurements. The latter means that measurements have been made only on data actually related to the Olympic Games.

Furthermore, analysis of user patterns and popularity prediction is done using the available data. In some cases, only data related to the Olympic Games has been used and in some cases all the available data has been used.

There is one point of particular interest in how the data that is generated by SVT Play looks. All programmes have an entry called `folderStructure`, which will basically say the name of the programme being streamed. If a live *rowing* stream was requested, the `folderStructure` entry would be `rodd.live`. This holds true for almost all types of programmes, except streams in the olympic games that were not of any particular sport (studio programmes, summaries etc.) or streams that contained many different events (such as the streams showing the content that was also televised). For some unknown reason, these have all been marked as *badminton*. For the purposes of not making measurements of e.g., popular sports inaccurate, they have been renamed first. Any programme marked as badminton that was not actually badminton, is instead categorised as *general*.

Chapter 5
Results

The different analyses and measurements that have been performed are explained in detail in this chapter. It is divided into four different sections: *general measurements*, containing some basic information gathered from all the SVT Play data and not only Olympic Games data; *Olympic Games measurements*, containing more detailed information about how people viewed the Olympic Games on SVT Play; *user clustering* in which it is explained how it is possible to place users in clusters based on their viewing preferences, potentially allowing for new methods of making smarter networks; and finally *caching*, in which it is presented what the optimal caching gain for this traffic measurement would be if an optimal cache was used.

5.1 General measurements

This section contains some general measurements done on all the data traffic during the olympic games period. It should be noted that while this is *general measurements*, it's still during a time period that generates atypical data to this streaming service. Normally, there is no such thing as Olympic Games live streaming and without additional data from measurements in the same area during different time periods it is difficult to know how the behaviour of users is different from how it normally is, e.g., there might be a lot more viewers than usually because the Olympic Games attract them, or the number of users might be the same as it usually is, but the distribution of categories different, *etc.*

5.1.1 Users online at different times

Information obtained from the connection log can be used to construct a graph that shows how the number of users watching streams varies over the day. The connection log shows shorter connections as well, e.g., when someone loads the front page or downloads an image. This is not really the kind of data that is interesting in this context. Therefore only connections with a length of over 10 seconds have been taken into consideration. Each IP address is only counted once at any given time. If a user is watching two streams simultaneously, it will still show up as one user. I.e., the graph shows the number of *unique* users online at a

given time. There may be more devices using a single IP at the same time, but it will be assumed that one IP has only one user.

The amount of users simultaneously online varies during the period. As expected, there is a lot of people streaming video content during the afternoon and evening. As can be seen in Figure 5.1, during the night and early morning, there's not a lot of people watching. Here, parallel streams have not been taken into consideration, but it is possible to have several streams running at the same time. At most, there are 56 different IP addresses online at the same time. No notable peak can be seen at 22:00 the first day when the opening ceremony started. This is probably because people watched the opening ceremony on their TVs or watched highlights or the entire ceremony as VoD. Sunday, 2012-08-05 was the day of the men's 100 m finals. The finals started 22:50 and that is the cause of the high peak of that evening which is also the highest peak during the entire time period. The

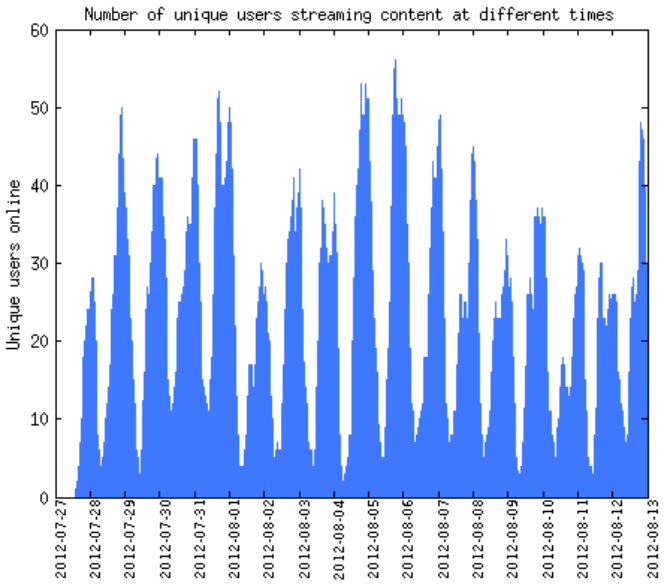

Figure 5.1: Simultaneous users online at different times during the entire Olympic Games time period.

graph pretty much follows what one would expect to see with regards to *when* people are accessing streams. Other studies done on different services (e.g., [20]) show this same behaviour, with peaks just before midnight and very little activity during late night to when people arrive at their workplaces. This effect might be augmented in this study because of the times during which these Olympic Games were broadcast, during the evenings.

5.1.2 Total number of viewers

There are some clients that generate traffic but do not start any streams, such as RSS readers. There are also some users that open the SVT Play website in a regular browser but don't start any streams. This means that counting the total amount

of users active during the olympic games period is not just a matter of counting the number of different MAC addresses that generated traffic. The selected method is instead to count the number of unique host destination MAC addresses, in response messages with `Content-type: application/json`. If such a message is sent, that's a sure way to know the client has started a stream. No other traffic measured will generate such a message. The size of the set of MAC addresses collected in this way is the number of viewers.

It may also be of interest to measure how many people viewed a certain programme. It is done in a similar fashion, but instead only a subset of users is counted, more specifically everyone who watched a show with that particular video ID.

Using the methods described above, it was found that 2085 unique users were, at one point or more during the entire time period, watching video stream on SVT Play. 1390 (66.67% of all users) of these watched at least something related to the olympic games. Of these, 915 (43.88% of all users) watched at least one *live* Olympic Games stream. This is a relatively small sample size. Making assumptions with such a small sample size can be dangerous, in an ideal world the sample size would be significantly larger, something that could possibly be achieved by doing measurements over a longer period of time, but that would defeat the purpose of doing the measurements during the Olympic Games. With such a small sample size the results should not be seen as decisively statistically significant, but of course not necessarily completely insignificant, either.

5.1.3 Popular stream categories

Each programme on SVT Play has a category that can be extracted easily from the JSON entity response messages. The JSON entity body has a key called `"statistics"`. The value is a list that contains the key `"category"`, amongst others. The category of the programme is written in clear, e.g., `"category":"barn"`. By counting each unique view, it is possible to see which category is the most popular. Since this is not necessarily live, it is possible to watch and re-watch the same content over and over. However a great majority of re-requests of the same content is due to browser refreshing or some clients sending requests over and over again at certain time intervals. Because of this, the method selected is to only count each user once per video ID.

The most popular category was by far the *Olympic Games*, with 9774 unique views. The second-most viewed category, *children's shows*, had quite a lot of views as well: 5541. The decrease is steep, however, and *culture* only had about a fifth of children's shows, at 1244 views. Figure 5.2 shows all the categories and their respective number of views. The categories are directly extracted from the response messages containing JSON files that are sent from SVT Play to the requesting client, i.e., no manual mapping of programme→category has been done. The category *Open Archive* contains old clips of "television history", pieces of television that SVT think is important and should be viewable by everybody and not be removed when the licence expires.

Again, it is quite evident that the olympic games is a popular event, with almost twice as many views as the second most popular category.

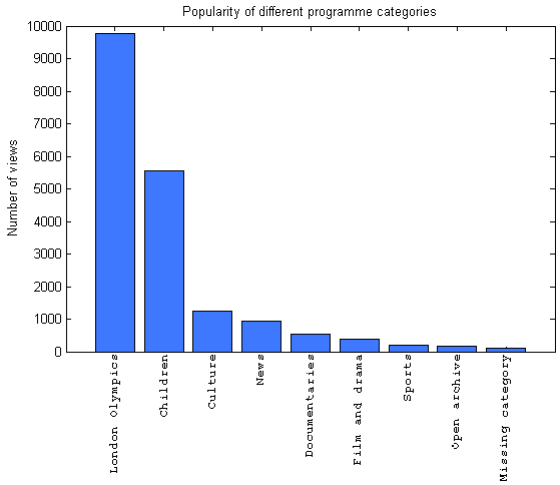

Figure 5.2: Categories on SVT Play and how many views each of them had.

5.1.4 Popular programmes

Looking further into the popular content than what category they belong to, it is of course possible to see the number of views per programme rather than category. The fifty most popular programmes are dominated by Olympic Games streams; 32 of them were streams from the events and a further few, e.g., *Mr Bean spelade i orkestern* (*Mr. Bean was playing in the orchestra*) from the opening ceremony, the programme about Usain Bolt, *The Fastest* and the documentary *Medaljens pris* (*The Price of the Medal*) can also be seen as being Olympic Games-related.

The most popular programme of them all was *OS Magasin: London Live med invigningen* (*Olympic Games Magazine: London Live and the opening ceremony*) at 131 views. When compared to the maximum number of users online at the same time in section 5.1.1, some contradition can be found. The numbers from this measurement is independent of time, there were never 131 users watching *at the same time*, which is why the peak number of simultaneous viewers is significantly lower than 131. Also, this is not the most popular live stream, since users could watch it even when it was no longer live. Not far behind the opening ceremony, at 123 views, comes the television stream from SVT1 during the last day, 2012-08-12. The stream contained mountainbiking for men, boxing, wrestling and marathon.

Kompisar på nätet (*A gURLs wURLd*) is a children's programme that was very popular and, in fact, the only children's programme to be in the top 50. A total of nine different episodes made it to the top-50 list, making this series a major contributor to the popularity of children's programmes.

All in all, 4305 different programmes were watched by all the users during the entire time period. Figure 5.4 shows the entire "top 4305" ranking without labels, to give an idea of how the views per video declines. If the long streams that contained more than one event is split up into several programmes there were a total of 4514 different programmes.

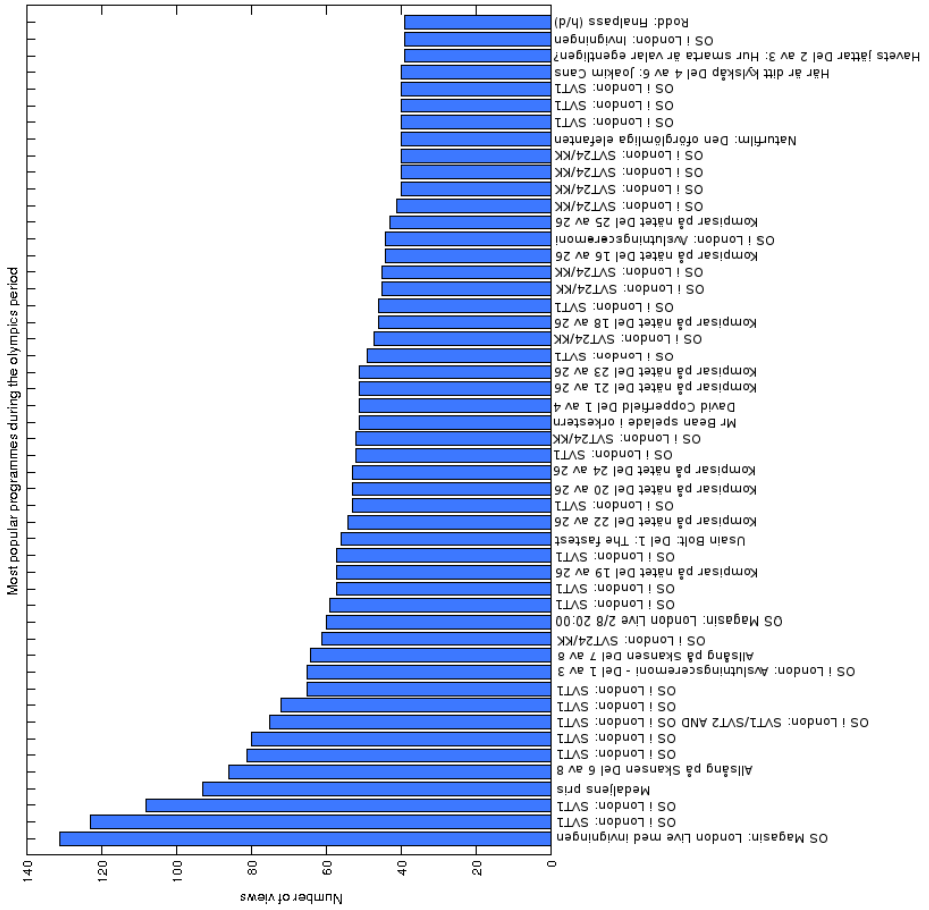

Figure 5.3: The most popular programmes on SVT Play. Top 50, sorted by popularity in descending order.

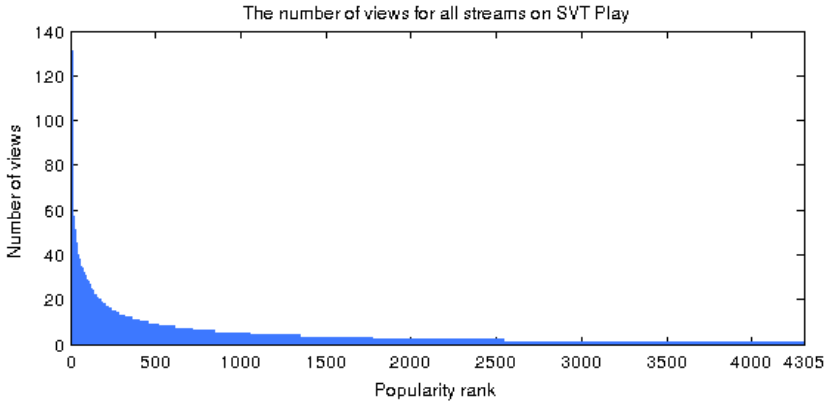

Figure 5.4: Views for all shows viewed on SVT Play. The most popular stream is at $x = 0$ and the least popular at $x = 4305$.

5.1.5 Programme views per user

The largest chunk of users by far, is the one containing those who watched only *one* programme. A histogram that shows how many people watched how many streams is shown in Figure 5.5. A lot of the users that only watched one stream (324 to be precise, see section 5.2.1) were people who only watched the Olympic Games, meaning the common one-show viewer is someone who tuned in to watch a specific event from the Olympics. With a total of 2 085 users and a total of 20 138 video views, both VoD and live, each user watched 9.7 different streams on average.

5.2 Olympic Games measurements

The main focus of this whole analysis is the London Olympic Games 2012. The opening ceremony that was the official start of the games was held on Friday, 2012-07-27 20:00 (UTC). There were two days with football games before the opening ceremony but they have unfortunately not been included in the measurements. The games ended with the ending ceremony which started 2012-08-12, 20:00 (UTC) and ended a couple of hours later.

While SVT Play allowed for users to watch VoDs the interesting data to analyse is that from the live streams. In the HTTP messages sent from the server there is a boolean parameter called `live`. The data used for the Olympic Games-specific measurements is data that has the `live` flag set to `true` and `category` to `os-london-2012`. There are many ways to see if a stream is live, e.g., the URL contains `/live/` and the programme name ends with `live`, or the `folderStructure` parameter ends in `.live`. All methods are equally good and give the same results. Only video views that meet these conditions are taken into consideration for Olympic Game measurements.

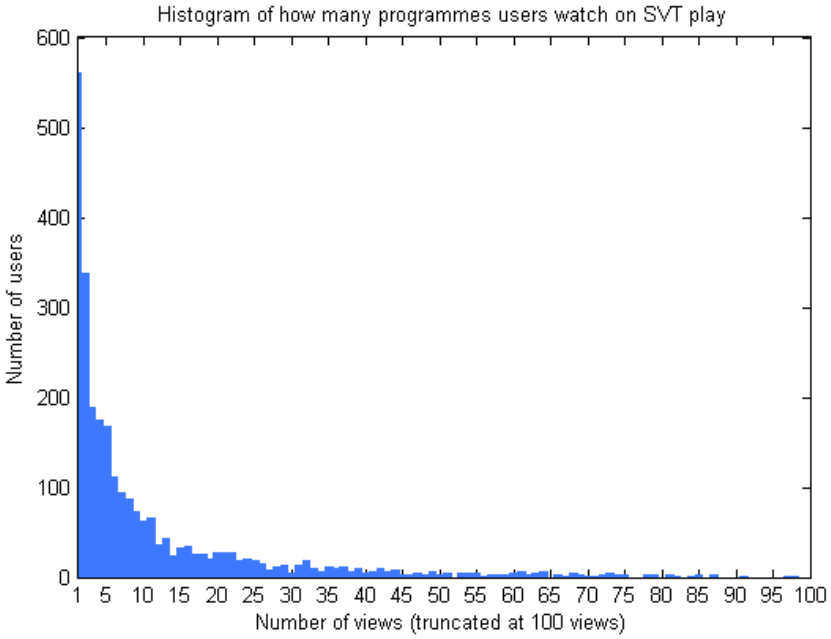

Figure 5.5: Histogram showing the number of different programmes a user watched on SVT Play. The largest chunk of users only watched one programme.

As a live online streaming event the Olympic Games were rather successful, getting a huge chunk of the total SVT Play stream views during the Olympic Games time period.

5.2.1 Popularity of live streams

Each programme, including live streams, has a video ID. The nature of a live stream is such that you can really only watch it once, even if you reload the stream it will still be the same session and the same programme. Therefore it's desirable to ignore any request from one MAC for a certain video ID after the first. By simply counting the number of MAC addresses that loaded a specific video at least once, it's possible to know how many users watched the stream. The problem with this method is that some streams are not tied to a particular sport and continuously broadcast a variety of content during the entire day while retaining the same video ID. These streams are generally the ones showing the same content as the televised channels. It is necessary to cut these up into smaller programmes and treat them all as separate programmes, which is done by extracting the TV times from an old-fashioned TV guide [23]. There is still a problem with this approach: a user might start the stream just a few moments before a certain sport starts being shown. The user can then watch the second sport for an hour, but the first request came during the first sport, and will count towards it in the statistics.

A total of 901 users watched the Olympic games. About a third of these, 324 users, watched only one stream. 2.0% of the views were by users with more than 40 views. The average amount of views per user was 6.3, but the median only 2.0. The much larger mean is mostly due to two heavy users with 127 and 164 views, respectively. Both of those users were very diverse in their preferences however and watched a lot of different sports. 5.6 shows a histogram of how many users had how many views. It has been cropped at 40 views, since the histogram is very scarce after that. Just like the histogram in Figure 5.5 that included all SVT Play data, it starts with a very steep slope and planes out quickly.

The 50 most viewed live streams during the specified time period are shown in Figure 5.7. One can see that the streams showing the same content as the television channels, *OS i London: SVT24/KK* and *OS i London: SVT1*, in the Figure shortened to just SVT24/KK and SVT1, are *by far* the most popular. Not only do they have the most views, they also dominate the list of top 50 most popular streams. The reason behind this ought to be that the most popular content was broadcast on TV.

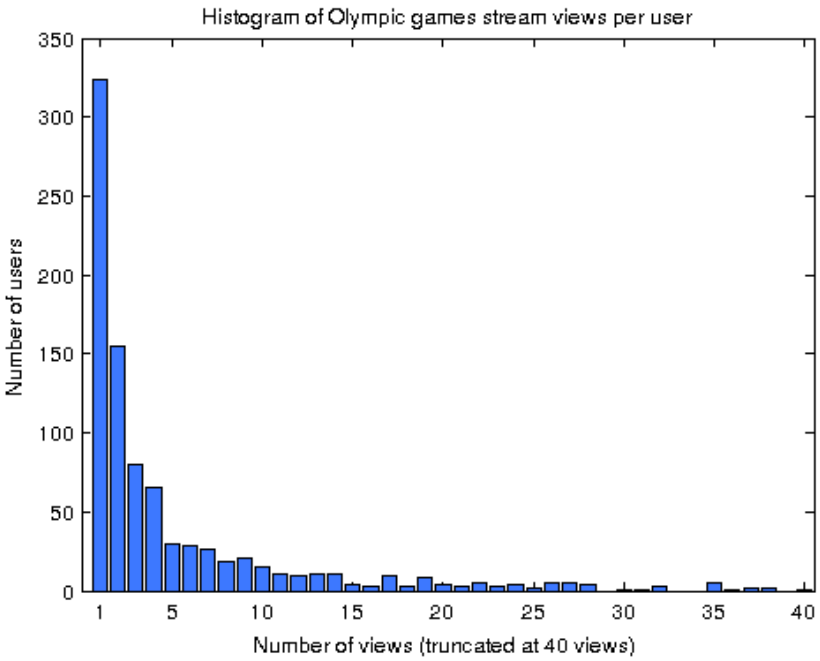

Figure 5.6: Histogram showing the number of different Olympic games live streams a user watched on SVT Play.

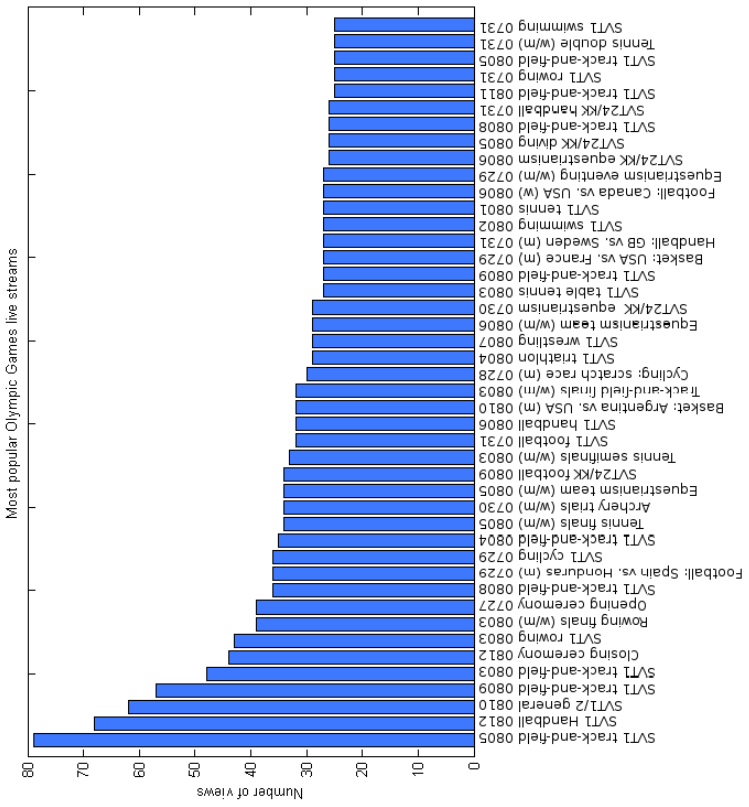

Figure 5.7: The most popular Olympic Games-related live streams and the date they were aired. "SVT" in the title means that the stream was showing whatever was currently on the television rather than a SVT Play-only live stream.

5.2.2 Popularity by sport

By doing further work on the list of programmes as defined above, one can extract the category, or in this case, *sport* of the streams. With this data it is easy to see which sports have been the most popular ones. There is one major flaw with this method that isn't related to the measurements in themselves, but rather the way the Olympic Games were broadcast. Most of the games took place during television prime-time in Sweden, during the evenings when most people are at home. The most popular sports, such as track and field, is of course what SVT showed on the TV. This means that while it was possible, there was no real need for people to watch online streams to access the most popular content. However those wanting to watch anything that wasn't televised would more or less have to rely on online streaming. This means that there is a possibility that the most popular sports are underrepresented and the less popular sports overrepresented in the collected online viewer statistics in relation to how popular the sports really were.

The events that the television broadcast company believe will be the most popular will of course be broadcast on TV (in addition to the online streams). However, they are the most popular events for a reason — people like watching them. This leads to the interesting question: does the fact that popular content is broadcast on TV mean it is not viewed as much online, or is it popular enough to have a lot of views *both* online and on TV? The answer can be seen in Figure 5.8.

As could be expected, the most popular sport is track and field with almost 200 more views than the second most popular sport, football. This is despite the fact that track and field was also shown on television, while many other sports were broadcast mostly online.

5.3 Prediction

Prediction of the popularity of VoD content is a problem that has been the subject for studies before. In [24], the popularity of content submitted on two sources of user-generated content, Youtube and Digg, is predicted. It is shown that very accurate predictions can be made after just a short "learning period". For the two portals analysed, it is also shown that the shorter the timelife of the content is the faster an accurate prediction can be made. It also shows that the more popular the piece of content is close to its time of submission, the more popular it will become later on. The evolution of submission popularity is described in the paper as such:

$$\ln N_s(t_2) = \ln[r(t_1, t_2)N_s(t_1)] + \xi_s(t_1, t_2) \tag{5.1}$$

where $N_s(t)$ is the popularity of a submitted content at time t, t_1 and t_2 two points in time such that $t_1 < t_2$ and $r(t_1, t_2)$ explains the linear relationship of the content's popularity at different times. ξ_s is a random additive noise term with mean 0 that describes randomness in the measured data. This is a very general description of course, and it has a weakness in that it can't predict if content will be popular before the content is actually uploaded. Also, the analysis was done on user-generated data.

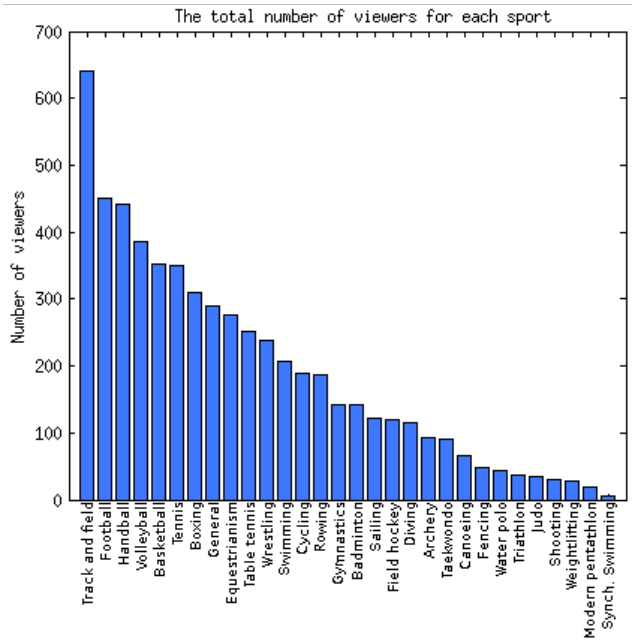

Figure 5.8: All the different sports shown live during the olympics and how many views each of them had. Beach volleyball and volleyball are both under the category "Volleyball".

In [25] it is examined how much data traffic can be saved by using a cache for popular content. A dataset from a VoD service of a "leading European telecom operator" is analysed. It is assumed every "item" is of equal size. Calculations are made on several different cache sizes. Several methods are used to calculate a theoretical optimal case of how much gain there can be using local caches. At first two classical methods are used, *Least Recently Used*, LRU and *Least Frequently Used*, LFU. In LRU the item that hasn't been requested for the longest period of time is removed from the cache. In LFU the frequency of requests for each particular item is stored. The items that are most frequently requested are put in the cache. A time window must also be specified, since a lot of requests too far back does not necessarily imply the content is still popular. Other methods are also examined, however these methods assume that a method for accurately predicting future requests already exists and is used together with the aforementioned two algorithms. This leads us back to the problem of actually predicting popular content before it becomes popular. In conclusion, LFU has a higher "cache hit rate" (probability of getting a request for an item that is already in the cache) than LRU, but it is also more complex and require a definition of e.g., the size of the time window. As the cache size grows, the difference in cache hit rate between the different algorithm gets smaller.

5.4 Popularity correlation

There may be reason to assume that two different sports can attract the same group of viewers. For example, football and handball are both team sports with similar goals (albeit different in their execution). Is it possible to say that if a user watches a lot of handball games, he or she is also inclined to watching a lot of football games? A very simple method to measure this is to use *Pearson's correlation coefficient* (see equation 5.2).

$$\mathrm{corr(X, Y)} = r = \frac{\sum_{i=1}^{n}(X_i - \overline{X})(Y_i - \overline{Y})}{\sqrt{\sum_{i=1}^{n}(X_i - \overline{X})^2}\sqrt{\sum_{i=1}^{n}(Y_i - \overline{Y})^2}} \tag{5.2}$$

If $\mathbf{X}$ is a column vector where the element X_i is the discrete value *football stream views* for user i and $\mathbf{Y}$ is a similar vector showing instead the number of views user i had for *handball*, we have two column vectors where each row represents one user and each column one sport. $\overline{X}$ is the expected value of random variable X, that is, $\overline{X} = \mathrm{E}(X) = \frac{1}{n}\sum_{i=1}^{n} X_i$. The correlation coefficient r can now be obtained using the above equation. r is upper and lower bound such that $r \in [-1, 1]$, a big positive value means high correlation while a big negative value means high negative correlation. $r = 0$ implies no correlation whatsoever [26].

The numerator of equation 5.2 is the covariance between $\mathbf{X}$ and $\mathbf{Y}$. The two factors of the denominator are the standard deviations of $\mathbf{X}$ and $\mathbf{Y}$, respectively. When the standard deviation of either of those is zero the resulting correlation coefficient is not defined. If the standard deviation of either is zero, that also means the covariance is zero. In that case the correlation coefficient is generally assumed to be 0 since both the numerator and denominator will equal 0. I.e., the assumption that $\frac{0}{0} = 0$ is made. But when calculating the correlation between a pair of vectors,

for example $\mathbf{X} = \binom{1}{1}{1}$, $\mathbf{Y} = \binom{1}{1}{1}$, having the correlation set to 0 seems counter-intuitive (how can a vector be completely uncorrelated to itself?). Therefore, it will instead be assumed that $r = 0$ if $s_x \neq s_y$ and $r = 1$ if $s_x = s_y = 0$, where $s_x = \sqrt{\sum_{i=1}^{n}(X_i - \overline{X})^2}$. While this is not an optimal solution to the problem, it works *well enough*. This is only a real problem if the sample size is very small, which is the unfortunate truth in these measurements. For the calculations, users with 0 views in both of the sports being correlated are omitted.

Positive correlation can tell us what consumers of a certain sport like, while negative correlation tells us the opposite; what they do not like. This coefficient depends, of course, on the samples taken during the specified time period. There is a chance correlation, if found, has actually manifested by chance rather than by existence of *actual* correlation or that actual correlation exists but it can't be measured.

Just measuring the correlation between sports is a very simple way of doing things and might or might not tell us anything about the reality whatsoever. The model of saying that the viewers of one kind of sport is a group is much too simple. By adding more sub-conditions to what a group is, this method might still be usable for calculating correlation. Sub-conditions such as *Sweden is one of the competing teams* and *the game is played after 17:00* might give results closer to the reality.

When measuring correlation values it is also important to measure the *p-values* of the obtained coefficient. The p-values tell the grade of *statistical significance* of the obtained correlation coefficients. A correlation coefficient can be high but if the measurement is not statistically significant the correlation value has no meaning. Reversely the correlation value can be low, implying no correlation at all, and still be statistically significant. Simplified, one could say that the p-value is the chance of obtaining the "wrong" value by random chance. Thus, a low p-value implies significance. The p-value is affected by such parameters as how much alike two vectors are and the size of the sample.

Table 5.1 shows the correlation constant for each pair of sports. The corresponding p-values can be found in table 5.2. Each sport has been given an index number. The key can be found in table 5.3.

The results show a lot of negative correlations between different sports that may not necessarily be negatively correlated. The popular sports are, in most cases, positively correlated with most other sports. The reason for this might be that the unpopular sports are seldom watched together by one and the same user, leading to a lot of pairs that has one view for one sport and zero views for the other, which results in negative correlation values. The most notable results include track and field, which is positively correlated to all other sports *sans* equestrianism and synchronised swimming. Other sports that have positive correlations to most other sports are football, boxing, handball, *etc.* The p-values are generally small for these popular sports with positive correlation. Results notable because of the opposite also exist; gymnastics, sailing, synchronised swimming and taekwondo all have mostly negative correlation to other sports but the results are in many cases not significant, further strengthening the hypothesis that they appear as negatively correlated in these measurements simply because they have not been

watched enough times.

The sample size and the short time interval make these results very unreliable and while they may not necessarily be completely wrong, it has already been established the great majority of users only watched one or a few programmes on SVT Play. Figure 5.9 shows a map of two sports and the amount of streams of that category viewed per user. The counters in the fields represents the number of users who had that many views of the two sports on the axes. This is a very typical plot for a pair of sports in the obtained dataset. Since users usually only watch very few programmes there is not many users that watched a lot of streams for *one* sport, much less two. Therefore the negative trend (negative correlation) is not uncommon, something that can most likely be explained with the very limited amount of data available rather than one sport being unpopular with a fan of another sport.

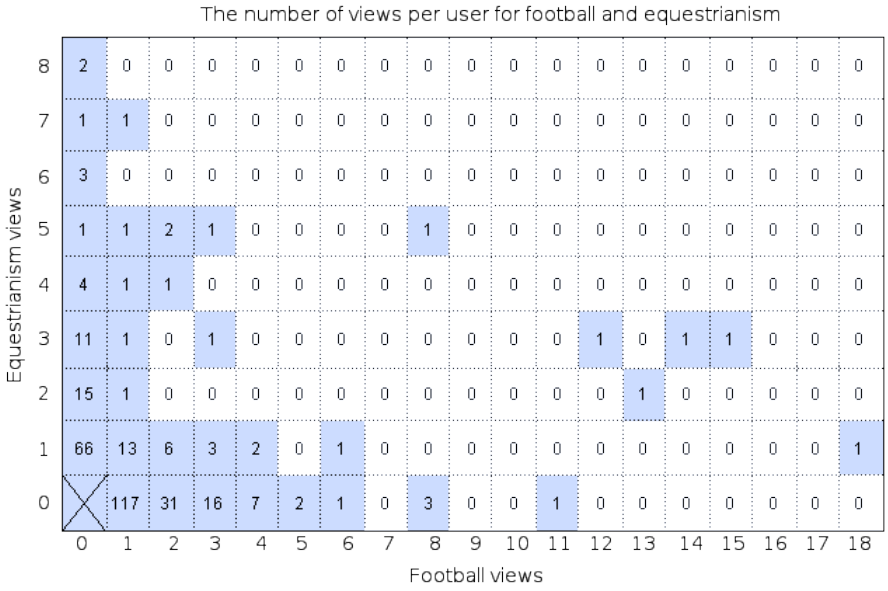

Figure 5.9: Equestrianism and football map. Each user who watched at least one stream of either football or equestrianism live is represented in this map. The counters in each box represents the number of users with the amount of views for each sport shown on the axes. A slight downward trend can be observed, there are no users with a lot of views for *both* sports.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
1	1.00	0.18	-0.17	0.02	-0.04	-0.22	-0.06	-0.14	-0.02	-0.22	-0.45	-0.08	0.29	0.12	-0.33	-0.17	0.05	-0.31	-0.14	-0.37	-0.41	-0.46	0.17	-0.19	0.21	-0.45	-0.17	0.18	-0.33	0.07	-0.32
2		1.00	-0.05	0.18	0.11	-0.03	0.06	-0.01	0.41	-0.03	-0.22	0.06	0.46	0.37	-0.08	0.25	0.23	0.05	0.21	0.08	0.07	0.13	0.10	-0.10	0.55	-0.06	0.05	0.40	0.33	0.35	0.16
3			1.00	-0.03	-0.16	-0.19	0.04	-0.18	-0.10	-0.39	-0.50	-0.09	0.29	0.14	-0.30	-0.21	0.00	-0.39	-0.31	-0.45	-0.35	-0.43	-0.02	-0.29	-0.05	-0.47	-0.15	0.08	-0.17	0.18	-0.40
4				1.00	-0.07	-0.06	0.00	-0.13	0.23	-0.14	-0.17	-0.07	0.19	0.21	-0.01	0.05	-0.06	-0.07	0.09	0.00	0.03	0.00	0.04	-0.16	0.15	-0.05	0.14	0.25	0.07	0.23	0.10
5					1.00	-0.18	-0.01	-0.10	0.29	-0.29	-0.24	0.12	0.24	0.10	-0.18	0.33	0.17	-0.25	-0.14	-0.33	-0.05	-0.07	0.39	-0.07	0.11	-0.28	-0.11	0.11	0.02	0.19	-0.16
6						1.00	0.03	0.02	-0.06	-0.30	-0.38	-0.00	0.13	0.17	-0.20	-0.32	-0.00	-0.26	-0.01	-0.13	-0.25	-0.36	-0.24	-0.13	-0.01	0.08	-0.03	0.10	-0.29	0.13	-0.14
7							1.00	-0.04	0.09	-0.17	-0.06	-0.04	0.28	0.17	-0.04	-0.07	0.21	-0.11	-0.09	-0.08	0.04	-0.04	0.08	-0.26	0.05	-0.07	-0.01	0.30	0.08	0.16	-0.05
8								1.00	-0.11	-0.23	-0.20	-0.14	0.11	-0.06	-0.02	-0.20	-0.06	-0.02	-0.12	-0.30	-0.15	-0.21	-0.15	-0.27	0.01	-0.11	-0.10	0.12	-0.11	0.05	-0.25
9									1.00	-0.25	-0.34	0.11	0.27	0.37	-0.15	0.29	0.04	-0.14	0.04	-0.12	0.01	0.10	0.19	-0.10	0.35	-0.23	0.08	0.23	0.34	0.35	0.11
10										1.00	-0.47	-0.20	0.14	-0.21	-0.18	-0.43	-0.00	-0.19	-0.20	-0.45	-0.57	-0.47	-0.33	-0.26	0.06	-0.38	-0.11	0.09	-0.44	-0.02	-0.51
11											1.00	-0.32	0.15	-0.23	-0.51	-0.50	-0.23	-0.91	-0.39	-0.83	-0.84	-0.86	-0.35	-0.23	-0.20	-0.92	-0.21	-0.10	-0.78	-0.10	-0.76
12												1.00	0.25	0.21	-0.05	-0.02	0.04	-0.04	0.05	-0.24	-0.02	-0.05	0.14	-0.19	0.06	-0.08	-0.09	0.25	-0.04	0.15	-0.13
13													1.00	0.32	0.04	0.17	0.40	0.18	0.21	0.21	0.16	0.07	0.19	-0.05	0.40	-0.08	0.20	0.46	0.18	0.46	0.13
14														1.00	-0.13	0.16	0.21	-0.11	0.10	0.07	0.17	0.09	0.26	-0.11	0.22	-0.07	0.03	0.32	0.27	0.35	0.21
15															1.00	-0.30	-0.11	-0.75	-0.01	-0.36	-0.64	-0.68	-0.18	-0.11	-0.04	-1.00	-0.06	0.04	-0.46	-0.02	-0.48
16																1.00	-0.05	-0.50	-0.16	-0.47	-0.25	-0.11	0.23	-0.14	0.26	-0.41	-0.07	0.11	0.11	0.25	-0.09
17																	1.00	-0.19	-0.08	-0.10	0.04	-0.12	0.04	-0.20	0.22	-0.19	0.02	0.30	-0.05	0.27	-0.18
18																		1.00	-0.11	-0.61	-0.75	-0.83	-0.35	-0.22	0.08	-0.87	-0.08	0.08	-0.68	0.08	-0.70
19																			1.00	-0.07	-0.27	-0.25	-0.20	-0.09	0.20	0.32	0.06	0.34	-0.06	0.26	-0.14
20																				1.00	-0.67	-0.67	-0.28	-0.21	0.10	-0.55	0.01	0.04	-0.52	0.19	-0.17
21																					1.00	-0.45	-0.08	-0.28	0.10	-0.57	-0.15	0.16	-0.26	0.08	-0.44
22																						1.00	-0.08	-0.19	0.20	-0.75	-0.09	0.11	-0.19	0.08	-0.40
23																							1.00	-0.04	0.07	-0.26	-0.13	0.20	0.02	0.21	-0.04
24																								1.00	-0.05	-0.09	-0.18	-0.06	-0.09	-0.02	-0.19
25																									1.00	-0.10	0.07	0.46	0.32	0.32	0.19
26																										1.00	-0.05	0.04	-0.58	-0.05	-0.46
27																											1.00	0.11	-0.00	0.31	-0.11
28																												1.00	0.18	0.43	0.16
29																													1.00	0.28	-0.26
30																														1.00	0.28
31																															1.00

Table 5.1: A table showing the correlation between the number of views of different sports. $\mathrm{corr}(X, Y) = \mathrm{corr}(Y, X)$ and $\mathrm{corr}(X, X) \equiv 1$. Significant results are in grey.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
1	0.00	0.02	0.06	0.82	0.63	0.01	0.37	0.07	0.80	0.03	0.00	0.29	0.00	0.08	0.02	0.09	0.53	0.01	0.14	0.00	0.00	0.00	0.09	0.01	0.00	0.00	0.01	0.00	0.01	0.28	0.00
2		0.00	0.47	0.00	0.09	0.67	0.29	0.94	0.00	0.69	0.00	0.33	0.00	0.00	0.30	0.00	0.00	0.55	0.00	0.32	0.36	0.08	0.18	0.13	0.00	0.48	0.39	0.00	0.00	0.00	0.04
3			0.00	0.64	0.04	0.02	0.55	0.01	0.24	0.00	0.00	0.19	0.00	0.03	0.00	0.01	0.95	0.00	0.00	0.00	0.00	0.00	0.83	0.00	0.50	0.00	0.02	0.20	0.08	0.00	0.00
4				0.00	0.27	0.33	0.96	0.04	0.00	0.04	0.01	0.28	0.00	0.00	0.92	0.46	0.31	0.33	0.19	0.95	0.67	0.95	0.58	0.01	0.01	0.51	0.02	0.00	0.31	0.00	0.18
5					0.00	0.01	0.81	0.14	0.00	0.00	0.00	0.10	0.00	0.13	0.05	0.00	0.02	0.00	0.06	0.00	0.59	0.40	0.00	0.26	0.09	0.00	0.08	0.06	0.79	0.00	0.06
6						0.00	0.59	0.80	0.45	0.00	0.00	0.97	0.01	0.01	0.04	0.00	0.96	0.01	0.89	0.17	0.01	0.00	0.00	0.07	0.93	0.44	0.68	0.09	0.00	0.03	0.13
7							0.00	0.52	0.14	0.01	0.33	0.49	0.00	0.00	0.55	0.29	0.00	0.12	0.15	0.21	0.59	0.52	0.20	0.00	0.42	0.31	0.85	0.00	0.24	0.00	0.48
8								0.00	0.14	0.00	0.01	0.03	0.03	0.31	0.85	0.01	0.37	0.85	0.12	0.00	0.06	0.01	0.04	0.00	0.86	0.16	0.08	0.05	0.17	0.41	0.00
9									0.00	0.00	0.00	0.13	0.00	0.00	0.14	0.00	0.55	0.17	0.62	0.21	0.93	0.30	0.03	0.17	0.00	0.02	0.23	0.00	0.00	0.00	0.25
10										0.00	0.00	0.01	0.01	0.00	0.16	0.00	0.99	0.10	0.03	0.00	0.00	0.00	0.00	0.00	0.42	0.00	0.10	0.17	0.00	0.80	0.00
11											0.00	0.00	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.14	0.00	0.12	0.00
12												0.00	0.00	0.00	0.59	0.83	0.54	0.63	0.48	0.00	0.78	0.57	0.06	0.00	0.36	0.37	0.12	0.00	0.58	0.01	0.11
13													0.00	0.00	0.50	0.00	0.00	0.00	0.00	0.00	0.00	0.24	0.00	0.27	0.00	0.17	0.00	0.00	0.00	0.00	0.02
14														0.00	0.08	0.02	0.00	0.13	0.16	0.34	0.02	0.20	0.00	0.06	0.00	0.33	0.56	0.00	0.00	0.00	0.00
15															0.00	0.01	0.18	0.00	0.95	0.02	0.00	0.00	0.13	0.21	0.64	0.00	0.43	0.58	0.02	0.78	0.00
16																0.00	0.48	0.00	0.09	0.00	0.02	0.31	0.02	0.06	0.00	0.00	0.30	0.08	0.31	0.00	0.42
17																	0.00	0.02	0.28	0.19	0.60	0.13	0.58	0.00	0.00	0.02	0.76	0.00	0.57	0.00	0.03
18																		0.00	0.31	0.00	0.00	0.00	0.00	0.00	0.28	0.00	0.24	0.21	0.00	0.20	0.00
19																			0.00	0.49	0.01	0.01	0.02	0.20	0.00	0.00	0.37	0.00	0.56	0.00	0.17
20																				0.00	0.00	0.00	0.00	0.01	0.16	0.00	0.87	0.58	0.00	0.00	0.18
21																					0.00	0.00	0.44	0.00	0.18	0.00	0.03	0.01	0.06	0.21	0.00
22																						0.00	0.42	0.02	0.01	0.00	0.20	0.11	0.22	0.19	0.00
23																							0.00	0.55	0.30	0.02	0.05	0.00	0.83	0.00	0.68
24																								0.00	0.44	0.27	0.00	0.30	0.28	0.75	0.01
25																									0.00	0.20	0.21	0.00	0.00	0.00	0.01
26																										0.00	0.46	0.51	0.00	0.47	0.00
27																											0.00	0.04	0.98	0.00	0.10
28																												0.00	0.01	0.00	0.01
29																													0.00	0.00	0.06
30																														0.00	0.00
31																															0.00

Table 5.2: A table showing the *p*-values of the correlations from table 5.1. Significant results are in grey.

1	Canoeing	**11**	Triathlon	**21**	Judo
2	Table tennis	**12**	Cycling	**22**	Weight lifting
3	Diving	**13**	Track and field	**23**	Sailing
4	Volleyball	**14**	Tennis	**24**	Equestrianism
5	Rowing	**15**	"OS-testet"	**25**	Boxing
6	Gymnastics	**16**	Archery	**26**	Synchronised swimming
7	General	**17**	Swimming	**27**	Basketball
8	Wrestling	**18**	Modern pentathlon	**28**	Football
9	Badminton	**19**	Field hockey	**29**	Shooting
10	Taekwondo	**20**	Water polo	**30**	Handball
				31	Fencing

Table 5.3: Translation table for sports in tables 5.1 and 5.2.

5.5 User clusters

By using historical viewer data, it is possible to construct clusters of users. Users inside one cluster have similar preferences, if the clustering is done well enough it is possible to assume that the probability of one user in the cluster watching or not watching a programme is high or low. It does not say anything about *which* programmes will be popular though, so the problem of predicting what users will watch still remains. It might be possible to drastically reduce the number of predictions that have to be done if the predictions are done for each cluster rather than each user, assuming the clusters are well-made.

Graph clustering is a hot topic in information visualisation, as enormous networks with hundreds of thousands of nodes and edges is very difficult to illustrate in a readable fashion. As such, there exists many algorithms for clustering graphs such as M. L. Huang's and Q. V. Nguyen's *Fast Algorithm for Balanced Graph Clustering* [27], which creates a graph where each cluster is of similar size, and the *Chinese Whispers Algorithm*, as explained in [28].

5.5.1 Edge creation

The nature of a streaming service allows for users to connect to the server to download content. This means that a user can be connected to a set of content, but one user never accesses another user and content does not access content. When clustering users, the idea is to make a cluster of only users, not content. Thus a connection between different users needs to be made. In other words, we want to transform a bipartite graph into a regular graph. Figure 5.10 shows an example of a bipartite graph, where the left side represents users and the right side different content. If a user has accessed any content, an edge is drawn between the user and the relevant content. The transformation from bipartite graph to regular graph can be done in many ways. Figure 5.11 shows an example where the users from Figure 5.10 have been linked together with an edge if they share one or more content accesses. This is a very primitive way of doing it and if the

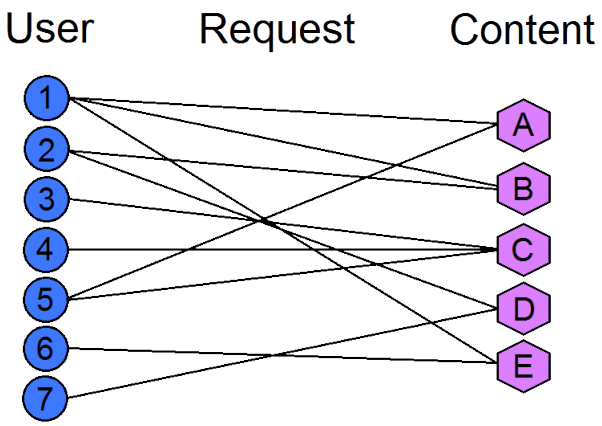

Figure 5.10: When clustering users, the first problem is to make a bipartite graph as shown in the Figure into a regular graph in which the users themselves are connected to each other rather than to content.

nature of the service is such that any one user with a very high probability will access certain content, the result will be that almost every user is linked together, which might or might not be what we want to do. To get nice clusters, it might be necessary to introduce some kind of threshold. For example, an edge will only be drawn between two users if they have both accessed five different live streams at the same time, or if some "similarity function" gives a high enough result for two nodes. The graph of users after they have been linked together in some fashion is

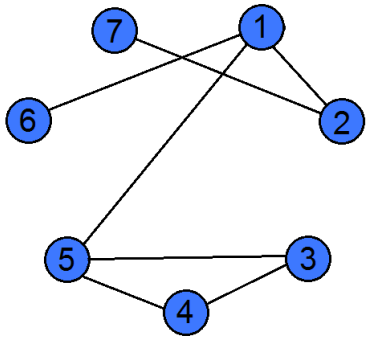

Figure 5.11: The users of Figure 5.10 have been connected to each other instead of the content they have watched. A simple form of edge creation has been used in this example; if two users accessed the same content they are linked.

denoted $G(\mathbf{V}, \mathbf{E})$, where $\mathbf{V}$ is the set of all users with nodes $v_i \in \mathbf{V}$. $\mathbf{E}$ is the set of all edges with $e_{ij} \in \mathbf{E}$, where the tuple $e_{ij} = (v_i, v_j, w_{ij})$ is an edge between two users v_i and v_j. Furthermore, w_{ij} is the *weight* of the edge. If the weight of every

edge in the graph is equal to one, the graph will be *unweighted*. If the weight can vary depending on how "strong" the connection between two users are the graph will be *weighted*.

General viewer edges

The experiments on the SVT Play data contains a few different types of graphs. Two ways have been selected to calculate the weight of the general data, that is, the set of every stream view, both VoD and live. Keep in mind that there can be an infinite amount of means to calculate this, no function that is specifically tailored to these traffic measurements has been used. Such a function would yield better results. The first way is to set the weight of every edge to 1, that is, *unweighted*. The second way is to set the weight of each edge to the number of programmes watched by both v_i and v_j. This is denoted *weighted*.

Olympic Games viewer edges

For the Olympic Games data the weighting has been done slightly differently. The edges for the unweighted case is the same as explained above, but instead of linking together users that watched the same programmes, the edges are drawn between two users that watched the same sport. The weighted case is slightly different. Instead of counting the number of different sports both users have watched, it also includes the number of programmes watched in each sport. If $V_s(v_i)$ is the number of views for sport s by user v_i, the weight of the edge is set to $\min(V_s(v_i), V_s(v_j))$. An additional third way of measuring weight, denoted *semi-weighted*, is used for the Olympic Games data. Semi-weighted has the weight set to the number of times a pair of users appear in a sport, doubles ignored. That is, the number of different sports they both watched once or more each, but each sport only counted once.

5.5.2 Chinese Whispers

One of the clustering algorithms, Chinese Whispers, is explained below. Originally an algorithm used for natural language processing, it can easily be applied on other data sets as well. Each node has a class, a cluster is the set of users having the same class. The neighbour of a node v_i is a node v_j between which an edge (v_i, v_j, w_{ij}) exists. The *neighborhood* of node v_i is every node it shares an edge with (though *not* v_i itself). The rank of the class in v_i's neighbourhood is the number of edges node v_i shares with a node of the class. The Chinese Whispers algorithm works as follows:

1. Draw the graph $G(\mathbf{V}, \mathbf{E})$.
2. Initialise by setting the class of v_i to i.
3. Go through each v in $\mathbf{V}$ in a random order. Set the class of each v to the highest ranked class in its neighbourhood. If there is more than one, pick one at random.
4. Repeat step 2 for the desired number of iterations.

The author of [28] has implemented a handy tool which executes the algorithm and shows a graphical representation of the clusters after a desired number of iterations. The tool can be found at [29]. By using the program on several example graphs (both on example graphs provided by the author and other third-party graphs) it can be observed that after zero iterations, every node lacks a class. After the first iteration, every node belongs to a random class. After the second iteration the clusters have already begun forming. After twenty iterations the clustered graph will usually be very similar to the graph after just two iterations. The sample graphs, packed with the tool, in Figures 5.12 and 5.13 illustrate this. The graphs show a large number of words from seven different languages. The clustering algorithm successfully divides words of each language into a separate cluster. A subgraph containing the French language after the clusters have been successfully created can be seen in Figure 5.14.

Figure 5.12: A sample cluster graph showing 1855 different words belonging to seven different languages. Two iterations of the Chinese Whispers algorithm.

While this clustering algorithm can indeed be used to find user clusters, it requires historical data from a user to be able to successfully place it in a relevant cluster. Using historical data it is possible to make *some* assumptions as to which live programmes will become popular, but it will not be able to detect it in real-time (unless the graph is updated in real-time). It can be used to some extent in order to make a network smarter, such as analysing categories instead of individual programmes, but that approach is flawed.

5.5.3 General data

The measurements on the general data shows excellent results. The unweighted graph can be seen in Figure 5.15. Two very distinct clusters can be seen: the *purple cluster* and the *light-blue cluster*. The purple cluster contains users that watched a lot of children's programmes, many of them watched *only* children's programmes. In the light-blue cluster, contains more diversified users, but the

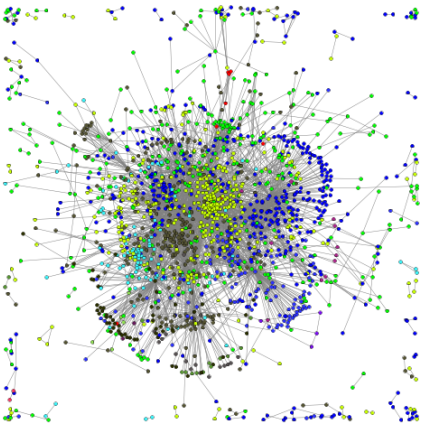

Figure 5.13: A sample cluster graph showing 1855 different words belonging to seven different languages. Twenty iterations of the Chinese Whispers algorithm.

Figure 5.14: A cluster of the graph in 5.12 and 5.13, showing the French language.

general trend is that they watch documentaries, factual programmes, news and community information programmes (debates, etc). The light-blue cluster is not as well-defined as the purple cluster is, in that the purple cluster is a very homogenous group with very little content that is not children's programmes. The results with

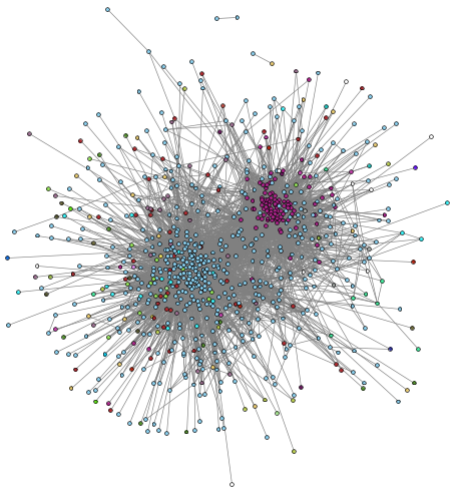

Figure 5.15: A graph showing two major clusters of users. The purple cluster contains users that watch children's programmes, while the light blue cluster contains a lot of factual programmes, documentaries, news and other programmes related to social matters. This is the results when running Chinese Whispers with unweighted edges.

unweighted edges can be seen in Figure 5.16. It more or less shows the same thing as the weighted graph, this is because, as mentioned, the children's programmes cluster is very, very homogenous and do not share many edges with users outside the own cluster. Had the light-blue cluster users been a bit more diversified, it is quite possible no distinct clusters would be found at all, or rather, that only one big cluster would be found.

5.5.4 Olympic Games data

The Chinese Whispers algorithm was executed on Olympic Games-specific data in the hopes of finding clusters of users with similar taste in sports. Unfortunately, no well-defined clusters were found using any of the aforementioned methods. As a matter of fact, all three methods gave edges that made the users converge into one large cluster. The results of clustering by sports could likely be improved by using a different way of going from bipartite graph to regular graph. The problem when all users are grouped into one big cluster like this, is that there is simply too many edges between users. While we do not want to remove edges arbitrarily, a measure of "likeness" or "similarity" between two users could be constructed. If they are similar enough, an edge is drawn between them. The method used has made too many of the users "similar" and thus must be improved.

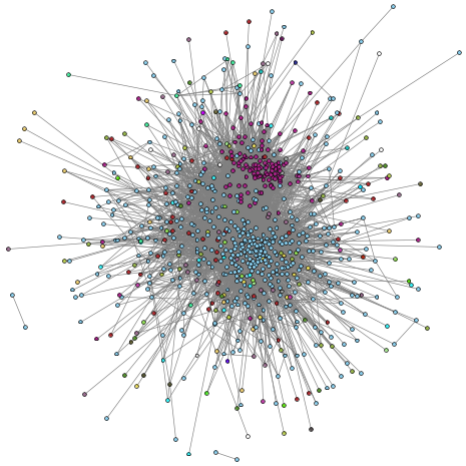

Figure 5.16: The same graph as 5.15, but this is the results when using weighted edges.

5.6 Caching

In this section it is described how an optimal mechanism for distributing both VoD and Live streams in one geographically limited network could reduce the amount of data traffic transported over the Internet. In both the cases the idea is to have some local mechanism that distributes the content to those who need it. The two cases are, however, a bit different from each other.

All the streams have been split into the two categories *Live* and *VoD* for the purpose of measuring cacheability, rather than *general* and *Olympics* as in most other measurements.

5.6.1 Caching in Video on Demand

It is assumed we have a magical box in our network that can save every video that is ever requested. Whenever a request to download a video arrives at the magical box, the box knows whether it has the content cached or not. If it does not, a request is sent to the original content server and the content is downloaded as usual. The data is then passed on to the original requester. If another user requests the same video at a later time, however, the magical box will already have it stored and can just respond to the requester with the video directly rather than having to further congest the rest of the Internet. For a conceptual sketch of how such a network could look, see Figure 5.17. The number of *items* that can be obtained (rather than bytes) from the cache is therefore equal to the number of unique videos that have ever been requested before. If N_0 users request the same item that item will only have to travel through the Internet once instead of N_0 times, meaning the reduced amount of items transferred through the entire Internet will be $\frac{1}{N_0}$ of the original amount that is required with today's *stupid networks*. More generally, during an infinitely long time period with N requests

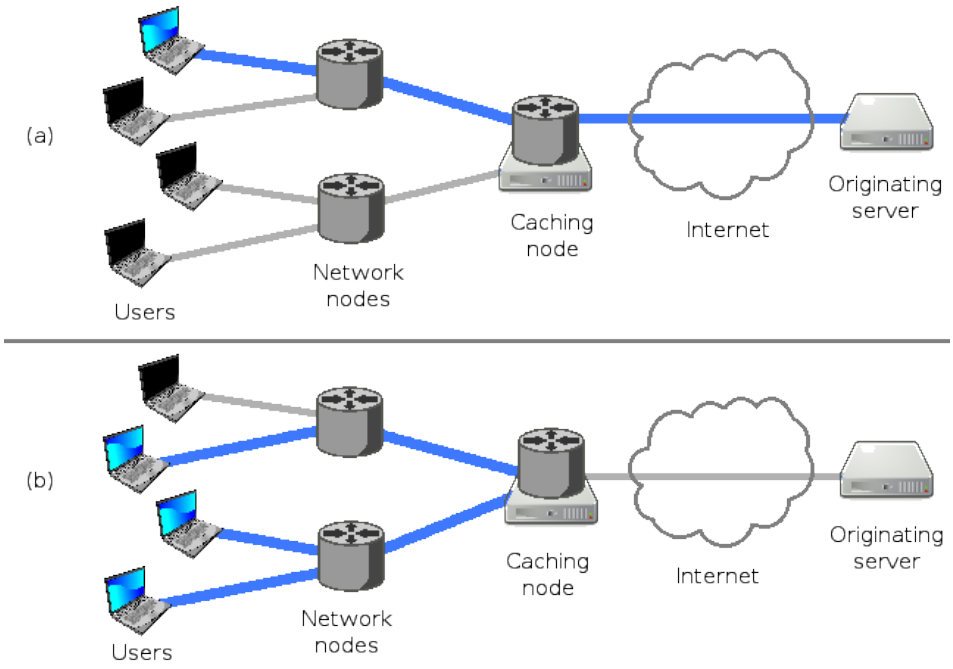

Figure 5.17: A network with a local cache. A thick line means a certain video is being transmitted on it. A thin line means that the video is not being transmitted on that connection. (a) When a certain video is requested for the first time, a copy of it is saved to a local cache in the network. (b) Subsequent requests for the same video will be transmitted to the users directly from the cache rather than going all the way back to the originating server.

for M different video streams the amount of items will be reduced to a factor $\frac{M}{N}$.

Assuming we start with an empty cache when the measurements started the final factor will be obtained by simply counting the number of requests that were not labelled `live`, N, and counting the number of unique videos requested, M.

The "cache hit ratio", or the probability of downloading a video that is already in the cache, will be $\frac{N-M}{N} = 1 - \frac{M}{N}$.

All in all, 13 619 different video on demand streams were started. The number of unique video on demand streams that were viewed was 3 711, giving the cache hit rate $1 - \frac{3711}{13619} = 0.73$, a very high probability. It would seem the users in the network are mostly interested in watching the same content!

5.6.2 Application-based multicast

Once again we have a magical box. This one does not store any data, however. This magical box knows which users are requesting which live streams. As long as there is more than one the request will have to travel through the entire Internet to obtain the stream from the originating server (as the nature of live video does not allow the network node to cache it before it is broadcast). The magical box can however send only one request to the server and then distribute it into its own network since it knows where the requesting users are. This is known as *application-based broadcast*. For a conceptual sketch of what such a network could look like, see Figure 5.18.

While the mechanism for distributing live streams in this fashion is totally different from the VoD caching example above the measurements can still be done in exactly the same way. No notice needs to be paid to the time of the request – at least not in this simplified example. Since a live stream is, by definition, only live while the event is still happening every viewer will be watching at the same time. Therefore, with N viewers for one live stream, only $\frac{1}{N}$ of the data needs to be transferred through the Internet. For M different live streams, once again the amount of *items* that require transport through the entire Internet is $\frac{M}{N}$. While the theory is the same as with cached content the probability of users streaming the same content can not be called cache hit ratio. Let us in stead call it "multiple live stream hit probability". That is, the probability that several users are watching the *same* live stream. The multiple live stream hit probability will be $1 - \frac{M}{N}$, just like the case with cache.

The number of different live events that were viewed was 803. Some of them were actually the same live streams, but they kept going for a long time and showed different events. Therefore they have been split up into events rather than just live streams. The 803 different programmes were viewed a total of 6519 times. The multiple live stream hit probability is therefore $1 - \frac{803}{6519} = 0.88$. That this would have a high value could almost be predicted beforehand, since the streams showing the same things as the television did were so popular. Most users that watched live streams watched them.

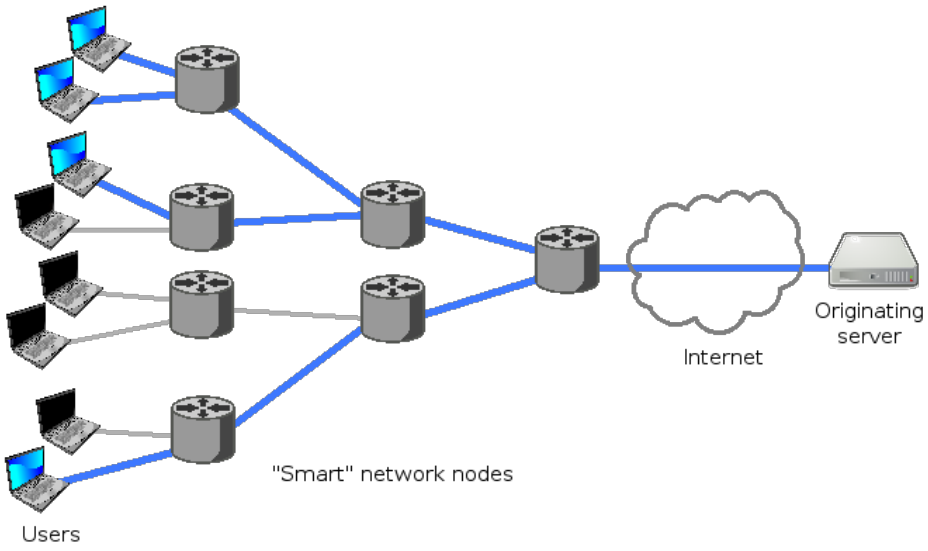

Figure 5.18: The originating server is broadcasting a live show that is watched by all blue users. The Internet edge will request the programme just once from the originating server and by some mechanism, distribute it to all the users in the network that want to watch it. The thick lines indicates that the programme is being transfered on that link, while the thin lines indicate that the programme is not transferred on that link.

Chapter 6
Conclusions

In this master's thesis I have proposed the use of clustering to divide users into different groups based on their interests and online video viewing patterns, both live and VoD. I have also analysed some of the basic statistics for SVT Play during the Olympic Games, this in order to be able to see some general user patterns. I have concluded that most users watched very few different programmes during the entire time period and that the most common type of user was the user who watched one stream from the Olympic Games. Furthermore, some theoretical calculations on how much bandwidth one could save on the Internet by using local caches or smart live video distribution networks were presented.

Measurements were done in a municipal network in a medium-sized Swedish city with many different types of users typical for a Swedish city. All measurements were done during the 2012 Olympic Games and the main purpose of this thesis was to see how users behave and how we can save bandwidth by observing the network during a period of intense Internet traffic, for which the Olympic Games was well-suited.

A general observation I have made when doing calculations and analyses of the traffic measurements, is that the sample size is much too small. More users during a longer time would be desirable, but analysis of the Olympic Games was one of the main points of this thesis, extending the time frame would not be feasible. Something that can be said about user behaviour is that users that watch a lot of children's shows watch little other programmes but the most popular so-called children's shows were shows for teenagers. Children's shows are popular because the young people of today live on the Internet and have Internet in their blood. I think that they don't see the need for the old-fashioned way of watching television, where the broadcaster and *not the user* decides when a show is on. Another explanation for the popularity of children's shows could be the use of "digital baby-sitters". When parents are busy cooking, cleaning the house, watching grown-up TV, SVT Play is readily available on the computer or the tablet, just start the kid's favourite show and he or she will do fine alone.

All the methodology, though perhaps with some modification assuming not all streaming services look the same under the hood, could be used for measuring similar statistics for other services. The clustering algorithm is very general and can connect two nodes of anything more or less, it just needs something to define when two nodes are connected, something that should probably be done in a unique way for each unique application. The calculation of caching gain could be

improved, for example by counting the number of bytes transfered instead of the number of items (or programmes) and by not assuming the cache is unlimited (of course no such device exists).

It has been shown that a few programmes has a lot of users and that the number of views per programme decreases rather quickly. The most viewed programmes, both live and VoD, were from the olympic games but children's programmes had a large share of the VoD segment as well.

Users were successfully clustered into different groups when using a primitive way of connecting two users when taking both live and VoD content into account, distinct clusters of children's shows viewers and more society-related programmes in a different cluster. The preferences of users in one cluster are similar, with an arbitrary measurement of similarity depending on how the users themselves are connected to each other in the first place. Unfortunately, no such clusters could be observed when looking only on data from the olympic games, most likely because a lot of users only watched one programme and because the users watching the olympic games were too similar.

6.1 Future work

There is a lot of room for improvement in the methodology. Some particular topics of interest are mentioned in this section.

6.1.1 Graph edge construction

In the clustering algorithm used, Chinese whispers, a bipartite graph is used to generate a regular graph. The methods used for the generation in this thesis work have been very simple but still shown some positive results. If a function for transfering a graph from bipartite is constructed to suit the application better the resulting clusters will be even better. Instead of simply counting the number of edges two nodes share, functions that take into consideration which types of content, popularity of viewed content and maybe other factors could be constructed. For example: if two people watch the opening ceremony that does not necessarily mean they have similar taste – a lot of people watched that stream. If however two people watch some obscure, sub-cultural programme that almost no-one watched, they would probably have pretty similar taste in that specific area, meaning the two users are more "alike" than two who happen to watch the opening ceremony. Factors like these have not been taken into concideration in this thesis, but such functions exist for e.g., social graph clusters.

6.1.2 Real-time clustering

In [30] the authors try to pick up hot trends and topics on blogging sites by making clusters of *key words*. When a group of two or more words appear together frequently, they will be correlated and eventually form a cluster. The clusters are only temporary, after the *blogosphere* loses interest in the topic, key words will appear together less frequently. This in turn means those clusters will disappear. In [31] it is argued that the method in [30] is not sufficient for real-time analyses of

trending topics as the graphs need to be constructed at certain time intervals and do not change dynamically. Therefore [31] proposes a new approach of clustering keywords dynamically, adding and removing both nodes and edges in real-time depending on current trending topics on the microblogging website *Twitter*.

This method is much more complicated than Chinese Whispers. It uses text streams such as Twitter to generate the graphs in real-time. If the method of pairing words that occur together and forming edges between them is generalised or adapted in such a way that live streams (which generally do not appear in "pairs") can be linked together, clustering of a network's users could be done much more efficiently than when historical data must be analysed and insterest in a future or current live show possibly extrapolated.

6.1.3 Caching

Only the amount of *items* that could have been streamed from a local cache was counted in this thesis. The reality is, of course, not that simple. Not every item is of the same size and there is no such thing as unlimited storage space. Also the caching gain was only calculated for the entire network, assuming the cache is at the Internet edge. This may or may not be the case in a real-world application. Many algorithms, as mentioned earlier, exist for caching, e.g., LFU and LRU. A lot of different ways of caching could be measured: geographically, with a certain cache size, with a certain caching algorithm and with different parameters for the caching algorithm used.

References

[1] *Quantities and Units – Part 1: General*, ISO 80000-1:2009, 2009.

[2] *Data elements and interchange formats – Information interchange – Representation of dates and times*, ISO 8601:2004, 2004.

[3] D. Graffox. (2009, Sept). *IEEE Citation Reference* [Online]. Available: `http://www.ieee.org/documents/ieeecitationref.pdf`

[4] R. Fielding et al., *Hypertext Transfer Protocol – HTTP/1.1*, IETF RFC 2616, June 1999; http://www.ietf.org/rfc/rfc2616.txt

[5] "Cisco Visual Networking Index: Forecast and Methodology, 2011–2016", white paper, Cisco Systems Inc., May 30, 2012.

[6] "Ericsson Mobility Report – On the pulse of networked society", press release from Ericsson, Nov. 2012. Available: `http://hugin.info/1061/R/1659597/537300.pdf`

[7] J. Pitta. (2012-01-16). *Between a Smartphone and a Hot Spot* [Online]. Available: `http://newsroom.cisco.com/feature-content?type=webcontent&articleId=631216`

[8] *Korta fakta om SVT* [Online]. Available: `http://www.svt.se/omsvt/fakta/kort-fakta-om-svt`

[9] AP. (2012-08-01). *What NBC paid for US Olympic rights over the years* [Online]. Available: `http://summergames.ap.org/article/what-nbc-paid-us-olympic-rights-over-years`

[10] *Celtic-Plus – Towards a Smart Connected World* [Online]. Available: `http://www.celtic-initiative.org/`

[11] International Telecommunication Union Statistics, *Global numbers of individuals using the Internet, total and per 100 inhabitants, 2001–2011* [Online]. Available: `http://www.itu.int/ITU-D/ict/statistics/`

[12] QBrick. *Online Video Customers* [Online]. Available: `http://www.qbrick.com/customers.aspx` (Accessed 2013-02-07).

[13] F. Hoffsümmer. (2013-01-25). *Tekniska milstolpar och framgångar 2012* [Online]. Available: `http://blogg.svt.se/testbild/2013/01/tekniska-milstolpar-och-framgangar-2012/`

[14] A. Hebert. (2012-05-07). *Det rör på sig – videoformaten i SVT Play* [Online]. Available: `http://blogg.svt.se/testbild/2012/05/det-ror-pa-sig-videoformaten-i-svt-play/`

[15] A. Hebert. (2012-09-03). *Ny streamingteknik för SVT Play* [Online]. Available: `http://blogg.svt.se/testbild/2012/09/ny-streamingteknik-for-svt-play/`

[16] M. Kihl et al., "Traffic analysis and characterization of Internet user behavior," in *Int. Congr. Ultra Modern Telecommunications and Control Systems and Workshops*, Moscow, Russia, 2010, pp. 224–231.

[17] A. Aurelius et al., "TRAMMS: Monitoring the evolution of residential broadband Internet traffic," in *Future Network and Mobile Summit*, Florence, Italy, 2012, pp. 1–9.

[18] A. Aurelius et al., "Leveraging network and traffic measurements for content distribution and interpersonal communication services with sufficient quality," in *Int. Conf. on Transparent Optical Networks*, Stockholm, Sweden, 2011, pp. 1–4.

[19] J. Li et al., "A five year perspective of traffic pattern evolution in a residential broadband access network," in *Future Network and Mobile Summit*, Berlin, Germany, 2012, pp. 1–9.

[20] M. Du, "Analyzing Caching Gain in Small Geographical Areas in IP Access Networks," M.S. thesis, Comm. Sys. Dept., Information and Communication Technology, KTH Royal Institute of Technology, Stockholm, Sweden, 2012.

[21] J. Haberkamm and N. Unnervik, "Facebook User Behavior & Community Clustering Analysis Based on Content Demand Patterns," M.S. thesis, Dept. of Electrical and Information Technology, Inst. of Technology, Lund University, Lund, Sweden, 2013.

[22] M. Verhoeyen et al., "Optimizing for Video Storage Networking With Recommender Systems," in *Bell Labs Technical Journal*, vol. 16, iss. 4, 2012, pp. 97–113.

[23] *Kalender och tv-tider Sommar-OS 2012* [Online]. Available: `http://www.sommaros.se/tv-tider`

[24] G. Szabo and B. A. Huberman, "Predicting the popularity of online content," *Communications of the ACM*, vol. 53, no. 8, pp. 80–88, 2010.

[25] J. Famaey et al., "On the Merits of Popularity Prediction in Multimedia Content Caching," in *12th IFIP/IEEE Int. Symp. Integrated Network Management 2011*, pp. 17–24.

[26] G. Blom et al., "Beskrivande Statistik," in *Sannolikhetsteori och statistikteori med tillämpningar,* 5th ed. Lund, Sweden: Studentlitteratur, 2005, ch. 10, sec. 4, pp. 233–234.

[27] M. L. Huang, Q. V. Nguyen, "A Fast Algorithm for Balanced Graph Clustering," in *11th Int. Conf. Information Visualization (IV'07)*, Zürich, Switzerland, 2007, pp. 46–52.

[28] C. Biemann, S. Teresniak, "Disentangling from Babylonian Confusion – Unsupervised Language Identification," in *Proc. Computational Linguistics and Intelligent Text Processing, 6th Int. Conf. (CICLing-05)*, Mexico D.F., Mexico, 2005, pp. 773–784.

[29] C. Biemann. (2006). *Chinese Whispers User Manual* [Online]. Available: `http://wortschatz.informatik.uni-leipzig.de/~cbiemann/software/CW.html`

[30] N. Bansal et al., "Seeking Stable Clusters in the Blogosphere," in *Proc. 33rd Int. Conf. on Very large data bases*, Vienna, Austria, 2007, pp. 806–817.

[31] M. K. Agarwal et al., "Real Time Discovery of Dense Clusters in Highly Dynamic Graphs: Identifying Real World Events in Highly Dynamic Environments," in *Proc. Very large data bases Endowment*, vol. 5, iss. 10, 2012, pp. 980–991.

Appendix A

Response messages

The following is an example of the gzipped JSON response entity body that is sent from the SVT Play server to the user client when user requests a video. This particular user is live streaming the opening ceremony. Note how the "statistics" entries "statisticsUrl" and "folderStructure" would have us believe the opening ceremony is badminton. Note also how there is many entries for the same video. These are for the different video qualities and/or different streaming protocols.

```
1 {
2   "videoId": 166548,
3   "video": {
4     "videoReferences": [
5       {
6         "url": "rtmp://fl10.c90807.cdn.qbrick.com/90807/
              webb4_360p_s",
7         "bitrate": 320,
8         "playerType": "flash"
9       },
10      {
11        "url": "rtmp://fl10.c90807.cdn.qbrick.com/90807/
              webb4_360p",
12        "bitrate": 850,
13        "playerType": "flash"
14      },
15      {
16        "url": "rtmp://fl10.c90807.cdn.qbrick.com/90807/
              webb4_576p",
17        "bitrate": 1400,
18        "playerType": "flash"
19      },
20      {
21        "url": "rtmp://fl10.c90807.cdn.qbrick.com/90807/
              webb4_720p",
22        "bitrate": 2400,
23        "playerType": "flash"
24      },
25      {
26        "url": "http://hls1.91001-live0.dna.qbrick.com/91001-
              hls1/live/webb4_os/webb4_os_iphone.m3u8",
```

```
27          "bitrate": 1400,
28          "playerType": "ios"
29        },
30        {
31          "url": "http://geoip.api.qbrick.com/services/rest/
                qticket/svtplay.aspx?vurl=http://qstream-live.qbrick
                .com/91001live_webb4",
32          "bitrate": 364,
33          "playerType": "wmv"
34        },
35        {
36          "url": "rtsp://rtsp0.91001-live0.dna.qbrick.com/91001-
                live0/_definst_/webb4_mp4-a-v1",
37          "bitrate": 340,
38          "playerType": "mpeg4"
39        },
40        {
41          "url": "rtsp://qstream-mblive.qbrick.com/91001/mob_live/
                webb4_3gp-c-v1.sdp",
42          "bitrate": 192,
43          "playerType": "3GPP"
44        },
45        {
46          "url": "rtmp://fl10.c90807.cdn.qbrick.com/90807/
                webb4_thumb",
47          "bitrate": 10,
48          "playerType": "flash-thumbnail"
49        }
50      ],
51      "subtitleReferences": [
52        {
53          "url": ""
54        }
55      ],
56      "position": 0,
57      "materialLength": 7140,
58      "livestart": -4320,
59      "live": true,
60      "availableOnMobile": true
61    },
62    "statistics": {
63      "client": "svt-play",
64      "mmsClientNr": "1001001",
65      "context": "svt-play",
66      "programId": "000000",
67      "mmsCategory": "3",
68      "broadcastDate": "20120727",
69      "broadcastTime": "2200",
70      "category": "os-london-2012",
71      "statisticsUrl": "http://ld.svt.se/svt/svt/s?svt-play.os-
            london-2012.badminton.live.os-i-london-invigningen",
```

```
72      "title": "os-i-london-invigningen",
73      "folderStructure": "badminton.live"
74    },
75    "context": {
76      "title": "OS i London: Invigningen",
77      "popoutUrl": "/live/166548/os-i-london-invigningen?type=
            embed"
78    }
79  }
```