Traffic analysis of Internet user behavior and content demand patterns

By

Irina Sánchez Muriana

Department of Electrical and Information Technology
Faculty of Engineering, LTH, Lund University
SE-221 00 Lund, Sweden

Abstract

The study of Internet traffic is relevant in order to improve the quality of service of users. Being able to know which are the most popular services and the hours with most active users can let us identify the amount of inbound and outbound traffic produced, and hence design a network able to support the activity expected. The implementation of a network considering that knowledge can reduce the waiting time of users considerably, improving the users' experience in the web.

Analysis of users' traffic and user demand patterns already exist. However, the data used in these studies is not renewed, thus the results found can be obsolete and considerable changes would have happened. In this bachelor's thesis, it is studied the amount of inbound and outbound traffic produced considering different applications and the evolution when regarding previous and actual data has been taken into account. This would let us understand the changes produced from 2007 to 2015 and observe the tendencies nowadays. In addition, it has been analyzed the user demand patterns in the beginning of 2016 and it has been contrasted with previous results.

The evolution of traffic has shown changes in users' preferences, although their demand patterns are still the same as previous years. The results found in this thesis confirmed the expectations about an increase of streaming media Internet traffic; it was proved that streaming media traffic is the dominant total traffic, with Netflix as the major contributor.

Acknowledgments

This Bacherlor's thesis would not exist without the assistance of my supervisor Maria Kihl, who guided me and helped me in this project.

I want to thank my parents for always being there and for their support in my stay in Lund. Also, I want to thank Alex, for his advice, his tolerance during all this time and for being by my side.

Irina Sánchez Muriana

Contents

Popular Science Summary

This thesis work describes how users are using the Internet in contrast with previous years.

Every time a user interacts with the Internet (opens a Webpage, watches a video, downloads any data...) generates a flow of traffic, which can have two different directions: inbound traffic, that is traffic that comes into the network, or outbound traffic, that is traffic that goes out of the network. Internet traffic can be divided into different categories depending on the source that is generating it, helping us to understand which are the most popular applications.

In this thesis, you can find an evolution of the inbound and outbound flows of the different categories from 2007 to 2015. The evolution of Internet traffic tells us which are the most important applications in the today's traffic, hence why users use Internet.

Another type of study that helps us to understand how users interact with the Internet is the analysis of user demand patterns. These analyses provide us with information about for example the hours with more users connected to the network or how long these users are online.

We have analyzed the user demand patterns of the beginning of 2016. Finding which parts of the day have the major amount of Internet traffic and also an evolution of popularity of the most popular video games.

The studies performed show us that audio and video services are the major contributors to the total Internet traffic. These kinds of services produce more inbound traffic than outbound traffic; hence the network has an asymmetrical behavior with more inbound than outbound traffic. The major contributor to this kind of traffic is Netflix, this fact maked us think about the importance of studying how users interact with this applications, that is why in this thesis an analysis of the length of Netflix sessions have been done.

If we know how people are using the Internet, we can design a network that will be able to support the amount of traffic expected depending on the time of the day. That way we can improve the experience of Internet users considerably.

1. Introduction

This thesis project is performed in the framework of the department of Electrical and Information Technology, Faculty of Engineering, LTH, Lund University with access to a database provided by Acreo Swedish ICT AB containing data about recent traffic analysis.

The traffic over the Internet is changing. There are numerous publications in the context of Internet traffic measurements, but the few studies focused on the long-term traffic patterns evolutions aren't renewed using actual data. These studies show some variations concerning the amount of traffic related to different categories. Another important aspect to analyse is the users' demand pattern. Although the results related to this study have proved that the users' demand pattern is less likely to change over the years than Internet traffic, it is necessary to know if users in 2016 are still using the Internet as before.

The thesis aims to analyse the studies about traffic analysis and users' demand patterns and contrast the information found with an analysis of the actual data. Taking into account previous and renewed data it would be possible to do an evolution of Internet traffic from 2007 to 2015, enabling a 7 year characterization of flows.

Following are some research that the degree project should answer:

- Has the user demand changed its pattern in contrast with previous studies?
- Is there a change in the total network traffic concerning different applications not seeing before?

The thesis will be divided in 3 parts. First of all it will describe of some important concepts that should be explained before starting the analysis of the data: Application protocols, video-on-demand systems, caching and prefetching and network traffic analysis. Once the basis about Internet systems and data acquisition have been explained,the thesis summarizes previous works related to Internet traffic in order to see the results found from 2007 to 2011. Finally, taking into account these results, the thesis starts the analysis of actual data that will enable to continue the studies exposed before with renewed results.

PART I: Important concepts.

2. Application Protocols

One of the packet characteristics that will let us identify the source type of traffic we are measuring is the application protocol.

Application protocols are found in the application layer of the Transmission Control Protocol/Internet Protocol (TCP/IP), a suite of protocols used to interconnect computer networks. Protocols in this layer specify the interaction between a pair of applications. The layer supports the network application programming interfaces (APIs).

This section presents an overview of some of the most used protocols, in order to understand how they work and how we would be able to identify them in our study: Hypertext Transfer Protocol (HTTP), Real Time Streaming Protocol (RTSP), Real Time Transport Protocol (RTP), Bit Torrent (BT), Direct Connect (DC), File Transfer Protocol (FTP), Internet Message Access Protocol (IMAP), Session Initiation Protocol (SIP), Media Gateway Control Protocol (MGCP).

2.1. Hypertext Transfer Protocol (HTTP)

The Hypertext Transfer protocol (HTTP) is an application protocol for distributed, collaborative, hypermedia information systems.

It functions as a request/response protocol:

- A client sends an HTTP request to the server; where it includes information about the request method (desired action to be performed on the identified resource), an identifier of the resource, the protocol version used, request modifiers, client information and an optional message body.
- The server returns an HTTP response message; where it includes the message's protocol version, a status code (success or error code), server information, entity meta information and an optional message body.

The server provides HTML resources that can be identified by a Uniform Resource Locator (URL), which is a specific type of Uniform Resource Identifier (URI).

2.2. Real Time Streaming Protocol (RTSP)

The Real Time Streaming Protocol (RSTP) was designed to establish and control streaming media services between a client and a server. It does not typically transmit the streaming data and it is often related to a 'network remote control' for multimedia servers.

It is similar in syntax and operation to HTTP/1.1.

The basic steps involved in the process are:

- The client establishes a TCP connection to the server.
- The client will send to the server details of the session requirements; version of RTSP, the transport to be used for the data flow and UDP or TCP port information. In order to send this information, the headers used are DESCRIBE and SETUP.
- Once the transport parameters are settled, the client is able to start sending requests such as PLAY, STOP or PAUSE in order to start, stop or pause delivery of the data stream.
- The client is able to close the stream using the TEARDOWN header.

An OPTIONS request can be used to get the requests types that the server will accept.

2.3. Real Time Transport Protocol (RTP)

Real Time Transport Protocol (RTP) is used to deliver real time information, such as interactive audio and video. It is used in systems that involve streaming media in conjunction with RTSP.

RTP does not guarantee delivery; the sequence numbers included in RTP are used to reconstruct, at the receiver side, the sender's packet sequence.

RTP was primarily designed to satisfy the needs of multi-participant multimedia conferences, although applications such as storage of continuous data, interactive distributed simulation, active badge, and control and measurement may also find RTP applicable.

RTP consists of two closely linked parts:

- The real time transport protocol (RTP), used for the exchange of multimedia data.

- The real time control protocol (RTCP), which aims to monitor the quality of service (QoS) and to transmit information about the participants in an on going session.

An application typically run RTP over UDP so it can use its multiplexing and checksum services, however, RTP may be used with other underlying network or transport protocols.

RTP and RTCP use consecutive transport layer ports, when used over UDP. First the audio/video is digitized (playload or coded data), then the bit streams formed are encapsulated in RTP packets, with the necessary timestamps and sequence number, and finally in UDP and IP packets. Fig. 1 illustrates these packet modifications.

The header format is shown in Fig.2, and the following information can be found:

- V (Version)(2 bits): RTP version.

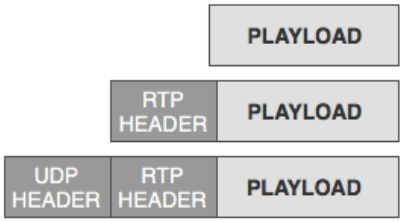

Fig. 1. Packet modifications

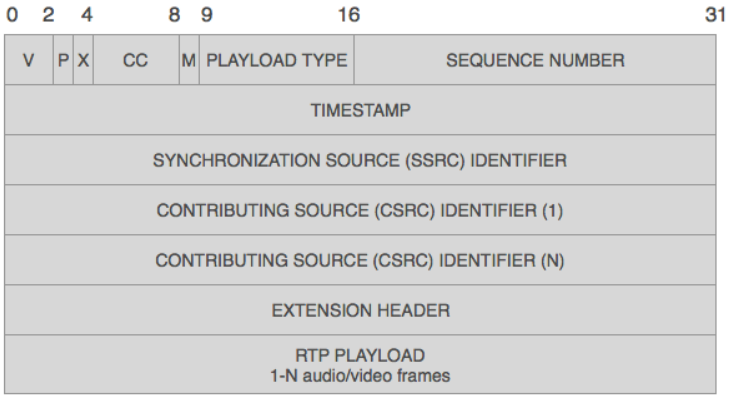

Fig. 2. Header format

- P (Padding)(1 bit): Indicates if there are extra padding bytes at the end of the packet.
- X (Extension) (1bit): Indicates the presence of an extension header between the standard header and the playload data.
- CC (CSRC count)(4 bits): Indicates the number of Contributing Source (CSRC) identifiers. This number is more than 1 if the playload contains data from several sources.
- M (Marker)(1 bit): It intends to allow significant events such as frame boundaries to be marked in the packet stream and its interpretation is defined by a profile.
- Playload Type (7 bits): Indicates the format of the RTP playload and determines its interpretation by the application. It is specified by a RTP profile.
- Sequence number (16 bits): Used by the receiver to restore packet sequence and detect packet loss. It is incremented by one for each RTP data packet.
- Timestamp (32 bits): Indicates the sampling instant of the first octet in the RTP data packet.
- SSRC (32 bits): Uniquely identifies the source of a stream. It indicates where the data was combined or the source of the data if there is only one source.
- CSRC (32 bits): Identifies the contributing sources for a playload, which has been generated from multiple sources.

2.4. Bit Torrent (BT)

BitTorrent (BT) is a protocol used for distributing files. It uses file swarming; where the BitTorrent protocol allows users to join a “swarm” of hosts to upload/download from each other simultaneously, also it allows the file source to support very large numbers of downloaders with a modest increase in its load.

A BitTorrent network is formed by the following parts:

- *Peers:* All the users in the network.
- *Leechers:* The users in the network who are downloading the file, they do not have it completely downloaded.
- *Seeders:* The users in the network who have the complete file.

- *Trackers:* An special server, which has the necessary information to establish connections between peers. It allows the location of the users who have the desire file to download.

To share a file, a peer has to create a file called a "torrent". It is a specially formatted binary file and has the following keys:

- *announce:* URL of the tracker
- *info:* this maps to a dictionary whose keys are the following:
    - *name:* suggested name to save the file if there is only one file or suggested directory name where the files are to be saved if there are multiple files.
    - *piece length:* number of bytes in each piece the file is split into.
    - *pieces:* a string whose length is a multiple of 20. It contains the hash of the piece at the corresponding index.
    - *length*: it is only present when one file is being shared. It contains the size of the file in bytes.
    - *files:* it is only present when multiple files are being shared. It contains a list of dictionaries containing the following keys:
        - *path:* a list of strings corresponding to subdirectory names, the last of which is the actual file name.
        - *length:* length of the file in bytes.

Once the file torrent is created, a client will act as a seed. Every user who wants to download the file has to obtain the file torrent. These users will start acting as leechers, sharing parts of the file with the seed and other clients.

Every time a peer gets a file piece, the hash of the piece is compared to the recorded hash to test that the piece has no errors. These file pieces typically are not downloaded sequentially, the packet sequence has to be restored at the client part.

Each leecher becomes a seeder by obtaining all the pieces, and assembling a file.

2.5. Direct Connect (DC)

Direct Connect (DC) is a text-based protocol used for peer-to-peer file sharing. In a DC network, there are three different components; hubs, clients and a superhub.

Hubs are the different servers, where each of them listens by default on TCP port 411. Every hub features a list of clients or users connected to it, hubs allow users to search for files to download or chat with other users on that server.

Clients discover hubs by asking the superhub, and all hubs register with the superhub.

The TCP data is a series of commands. Protocol commands start with $ and ends with | . A client must be logged in before sending or receiving any message.

2.6. File Transfer Protocol(FTP)

The File Transfer Protocol (FTP) is a protocol defined to transfer files between a server and a client.

In an FTP service, the user-protocol interpreter (user-IP) initiates the control connection. The user-IP generates standard FTP commands and they are transmitted to the server process. In response to the commands, the server-IP sends standard replies to the user-IP. Fig 3. illustrates the interactions between the different components in a FTP system.

The FTP commands specify the parameters for the data connection; data port, transfer mode, representation type and structure, and also the nature of file system operation; store, retrieve, append, delete etc. The user-data transfer protocol (user-DTP) or its designate should listen on the specified data port and transfer the data in accordance with the specified parameters.

2.7. Internet Message Access Protocol (IMAP)

The Internet Message Access Protocol (IMAP) allows accessing and manipulating electronic mail messages. The protocol assumes a reliable data stream as the one provided by TCP.

The actual version of IMAP is IMAP4revl. Its connection has the following steps: first of all, a client/server network connection has to be established, followed by an initial greeting from the server and finally

client/server interactions. The client/server interactions consist of a client command, server data and a server completion result response and they are in the form of a string that ends with a Carriage Return, Line Feed (CRLF).

Every client command is a new operation. A “tag”, which is an identifier, prefixes each command.

Server data is sent as a result of a client command or is sent unilaterally by the server, and in any case, it is prefixed with the token “*”.

The server completion result response indicates the success or failure of the operation, and there are three different possibilities: OK (success), NO (failure) or BAD (protocol error, such as unrecognized command or command syntax error). This response is tagged with the same tag as the client command, which indicates the operation.

2.8. Session Initiation Protocol (SIP)

The Session Initiation Protocol (SIP) allows establishing, modifying and terminating multimedia sessions or calls. It is designed to be independent of the transport layer.

SIP can initiate sessions and also invite members to sessions already established.

Callers and callees are identified by SIP addresses called SIP URLs. The SIP URL looks like, user@host; where the user part is the user name or a telephone number and the host part can be a domain name or a numeric network address.

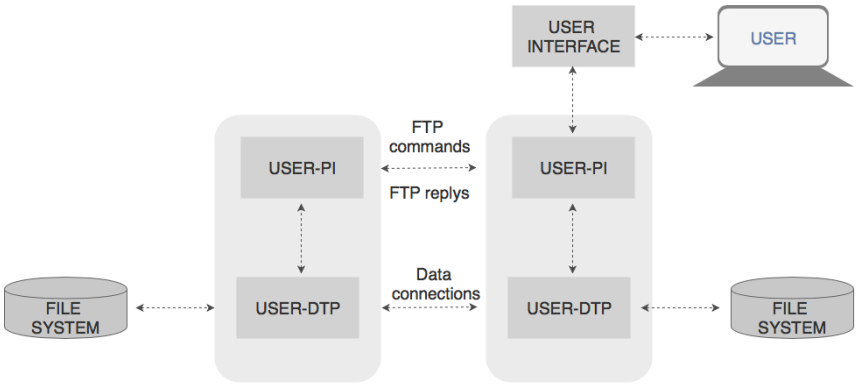

Fig. 3. FTP system

When making a SIP call, the caller first locates the appropriate server. It can be a locally configured SIP proxy server or the IP address and port corresponding to the Request-URI (it indicates the user or service to which the request is being addressed). Once the server is located, the client sends one or more SIP requests to it and receives one or more responses.

A request line and usually a body message form the SIP requests. The request line begins with a method, followed by the Request-URI and the protocol version, and ends with CRLF.

The SIP responses contain a status line and usually a body message. The status line consists of the protocol version, followed by a numeric Status-Code and its associated textual phrase. As the request line, the status line ends with CRLF.

2.9. Media Gateway Control Protocol (MGCP)

The Media Gateway Control Protocol (MGCP) controls media gateways from external call control elements called media gateway controllers or Call Agents. It is used between elements of a decomposed multimedia gateway.

A media gateway provides conversion between the audio signal carried on telephone circuits and data packets carried over the Internet or other packet networks. The Call Agent contains the call control "intelligence".

MGCP is a master/slave text-based protocol, where the gateways are expected to execute commands sent by the Call Agents. It assumes a connection model where the basic constructs are endpoints, which are sources of data, and connections.

Each endpoint has its own identifier composed by two different components: the domain name of the gateway managing the endpoint and a local name within the gateway (local-endpoint-name@domain-name).

The Call Agent on each endpoint of a call has to create the connections. There can be two different situations considering a connection between two endpoints; the two endpoints are located on gateways managed by the same Call Agent or the two endpoints are located on gateways managed by different Call Agents.

In the first situation the connection is established following three different steps:

1- The Call Agent asks the first gateway to 'create a connection' on the first endpoint. The gateway allocates resources to that connection, and responds to the command by providing a

'session description'. It contains the information necessary to send packets towards the new connection (IP address, UDP port and codec parameters).

2- The Call Agent asks the second gateway to 'create a connection' on the second endpoint. The 'session description' provided by the first gateway is sent to the second gateway. The second gateway allocates resources to that connection, and provides its own 'session description'.

3- The Call Agent uses a 'modify connection' command to provide the second 'session description' to the first endpoint.

Once the three steps are completed, the communication can start in both directions.

When there are two different Call Agents, the Call Agents exchange information through a Call Agent to Call Agent signalling protocol to synchronize the creation of the connection on the two endpoints.

A 'delete connection' command will be used for deleting an existing connection.

3. Video On Demand systems

As it will be seen in future sections, streaming Internet media traffic has been increasing considerably. One of the main reasons of this increase is the popularity of Video on Demand systems (VOD). VOD are systems that allow users to browse, select, and watch/listen to media content when they want to, from a large content repository on an on-demand basis, rather than having to watch at a specific time.

Due to the importance of those applications, this section aims to explain how these systems work. First of all it is necessary to know the components that form VOD systems, as described in section 3.1. In order to interconnect these components an architecture is needed and the different network architectures are found in section 3.2. In section 3.3 the requirements of a VOD system are presented in order to guarantee a good Quality of Service. And finally, it is also interesting to know how the data is provided from the server to the final users and this information is found in section 3.4.

3.1. VOD components

A VOD system consists on three different parts:

- *Client:* A client has to receive, process and visualize the video signal generated by the server. Also, the client has to ensure the synchronization of audio and video and acts as interface between user and server. Sometimes a client can also act as a server for other clients in the same neighbourhood. The client can be a home computer with a web interface or application, or a Set-Top-Box (STB). A STB is a device dedicated exclusively to video reception and decoding.
- *Server:* The server receives the client petitions and delivers the contents. The server is responsible for managing the different resources and guaranteeing the Quality of Service (QoS) the clients need.
- *Network:* The network connects servers and clients and composes the transport media. Delay must be minimized in order to preserve presentation. In the network, there can be continuous and long-lived connections.

3.2. Network Architectures

There exist various network architectures which aim is to minimize network costs and satisfy the QoS restrictions. But, as we don't need to know much about these architectures and we just need to have an overview in VOD systems, in this section descriptions of two of the most widely used architectures are presented.

- *Centralized architecture:* In a centralized architecture there are no local servers. Server and archives are stored in a central location. All the client hosts communicate with the network's primary servers through a broadband channel, which represents the backbone of the network. The primary server's router receives all the requests from the clients, acting as a gateway. Via the outbound links, the video primary servers transmit the video data information to the destination client. This architecture suffers from scalability problems and low throughput as the video traffic will be concentrated near the primary server, but the architecture is easy to manage. It requires large-capacity servers with good network connections in order to provide access to a large library of content. Fig 4. exemplifies a centralized architecture.

- *Distributed/Hierarchical architecture:* In a distributed architecture, there are allocated local proxy servers at strategic locations in the network, usually close to the clients. Client hosts can communicate with both servers; the network's primary server and local proxy servers. Customers are connected to the local server through a

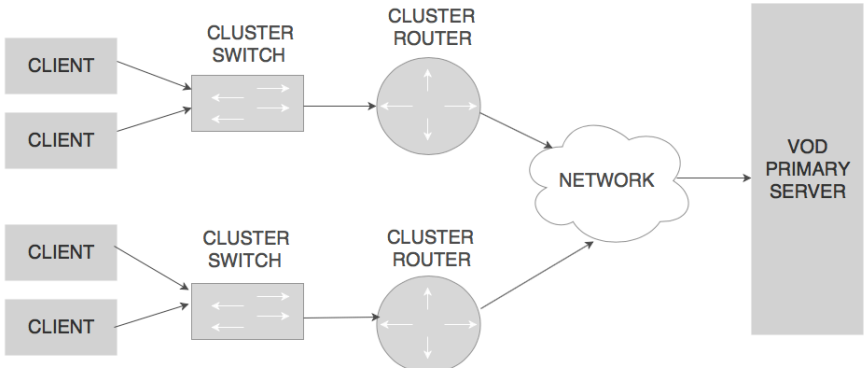

Fig. 4. Centralized Architecture

cluster switch and they are connected to the primary server through a cluster router. The request of a client will be only sent to the primary server if the local proxy servers cannot serve the user. This can take place, for example, if there is a blockage of the local server, or the multimedia information required is not available in the local server. This architecture is called also hierarchical because the requests are sent, when they cannot be processed, to the server of the upper hierarchical level, until reaching the primary server. For that reason, the most popular content is expected to be close to the user location. As can be seen, this architecture allows distributing the traffic within the network. This architecture allows a reduction in the delay and also in the load on the system. What is more, it does not have scalability problems as the centralized architecture; local servers can be included easily.

Fig 5. illustrates a one level hierarchical architecture.

3.3. VOD requirements

In order to provide a good quality of service, Video on demand systems need to specify some requirements concerning its performance, some of these requirements are:

- *Time sensitivity:* In a VOD application, the delay between sending a packet and the reproduction of the video/audio information contained in it may be of the order of seconds due to that streams are sent and received in real time. We can define two different metrics:

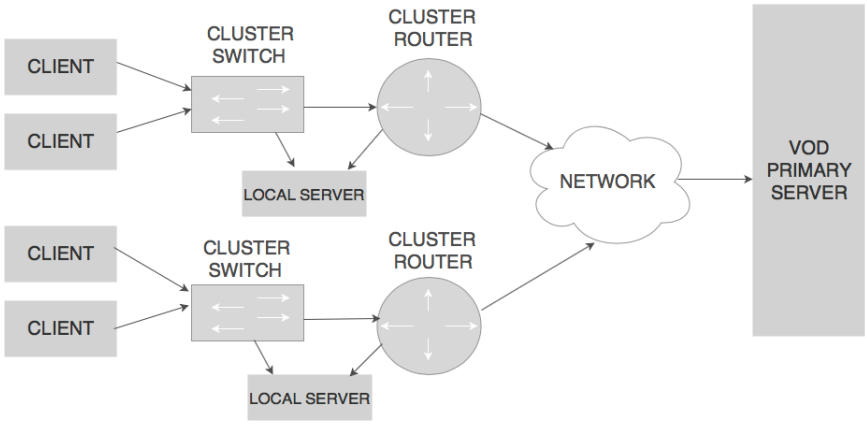

Fig. 5. One level hierarchical Architecture

    - *Start-up delay:* The start-up delay defines the waiting time from the moment when a client request a video until the moment when the user begins to see the video requested.
    - *Response time of the interactions:* The response time defines the latency from the time of transmitting a control command to the actual change of the display scene.
- *Streaming capacity:* The streaming capacity is defined as the number of concurrent users or number of commands/requests the server can handle per second (request rate).
- *Loss or blocking probability:* A block occurs when a request cannot be served immediately. It is essential to have aa low loss or blocking probability in order to provide a high QoS to the users.
- *Scalability:* It is important to design a system with facilities for future expansions in order to provide services to more users, that is, the system needs to be scalable.
- *Cost-effectiveness:* A VOD system should fulfil all the requirements with the lowest cost possible. This cost consists of many components such as communication costs, bandwidth costs, storage costs etc.
- *Other capabilities:* Depending on the type of VOD systems, more requirements will be necessary such as different video qualities or scheduling policies.

3.4. Data delivery

The easiest way to deliver the request content to a client in a VOD system is to send a dedicated stream to each client (Unicast Data Delivery). This approach increases the server and network load and can deplete server resources. In order to save resources on the server and network sides some stream aggregation techniques can be used (Multicast Datagram Delivery). These techniques assumes that the VOD systems support a great number of clients and requests of the same content in a short period of time are likely to happen. They are called aggregation techniques because servers aggregate client requests for the same file and serve them using a multicast channel.

There exist different aggregation techniques, some of them are:

- *Staggered Broadcasting:* Staggered broadcast is the simplest technique, where entire copies of a media file are repeatedly

broadcast on separate channel, each at the media playback rate. The media file is partitioned into *K* equally size segments. The duration of each segment is $s = D/K$, where *D* is the total duration of the media file. Given the media object *consumption rate r (bps)* and assuming that the server has enough bandwidth available, a server bandwidth of $r \times K$ is allocated. Then, the bandwidth is divided in order to get *K* equal channels, each of them will repeatedly broadcast the video with a *transmission rate r*. The *phase offset* is the difference in starting times and is the maximum time clients have to wait for their request to be served, as clients who request a video during a current *phase offset* will be grouped together to watch the next broadcast. That way, clients only has to listen to a single channel to receive an entire video. The distribution of the media file segments into the different channels is shown in Fig 6.

- *Pyramid Broadcasting:* In Pyramid Broadcasting, instead of dividing the media file in *K* equal size partitions, the file is divided in *K* partitions of increasing sizes following the function in (1). The available bandwidth for each video, *B*, is divided into *K* channels, producing a *broadcast rate* at the server of *B/K* on each channel. Each channel broadcast repeatedly one segment of the media file. The *waiting time* before the video is started to be played equals the *waiting time* until the first segment is downloaded. Once the clients start to consume S1, they can begin to receive S2 on channel C2 and so on.

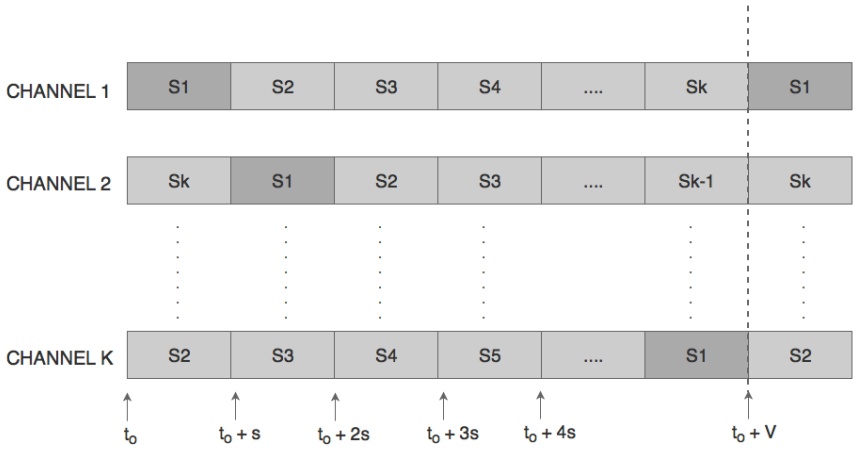

Fig. 6. Staggered Broadcasting segments distribution

The small first segment permits low *start-up latencies* while the larger later segments allow a reduction in the total number of channels needed for the broadcast. A disadvantage is that the *transmission rate* on each channel must be much higher than the *media playback rate* to guarantee the arrival of each segment by the time the current segment finishes playing. This produces the necessity of storing a large amount of data, and therefore this technique requires caching at the client side. The storage has to be large enough to hold almost the entire last two segments of the media file, and these two segments typically amount for 75-80 % of the total size.

$$S_i = \begin{cases} \dfrac{V(\alpha - 1)}{\alpha^k - 1}, & if\ i = 1 \\ s_1 \times \alpha^{i-1}, & otherwise \end{cases} \quad (1)$$

- *Permutation-Based Pyramid Broadcasting:* Permutation-based pyramid broadcasting proposes some modifications to the Pyramid Broadcasting technique. Each channel is partitioned into *P* subchannels with *KB/P bps* each. A replica of segment Si is repeatedly broadcast on the *P* subchannels of channel *i* and the starting times of each replica are uniformly staggered with a delay of *Si/P* seconds. This reduces the client storage space requirement to 50% of the media file size. In order to understand this performance better, a simple comparison between pyramid broadcasting and permutation-based pyramid broadcasting can be found in Fig. 7. In Pyramid Broadcasting, when the client has finished to download segment Si, the *play time* is shorter than the time the client has to wait until downloading segment S1+1 plus its completely download. For that reason, it would be necessary to have segment S1+1 already in the buffer. In Permutation-Based Pyramid Broadcasting, when the client has finished to download Segment Si, the client can start and finish the download of segment Si+1 before ending the *play time* of the segment Si. As can be seen, it is not necessary to have segment Si+1 already in the buffer.

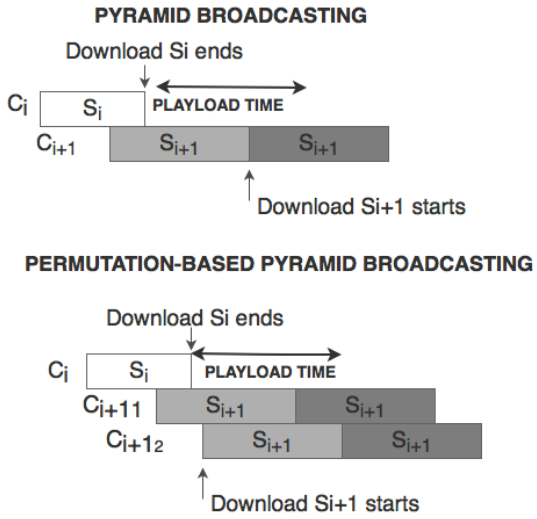

Fig. 7. Pyramid Broadcasting and Permutation-Based Pyramid Broadcasting comparison

- *Skyscraper Broadcasting:* Skyscraper Broadcasting also divides the server bandwidth into *K* equal channels each of *rates B/K*. In every channel a segment of the video will be broadcasted repeatedly. The difference is found in the way the file is decomposed, the function is shown in (2). As can be seen, the sized of the segments are *[1,2,2,5,5,12,12,,25,25,52,52...] x S1*. This series determine the number of consecutive segments to place on each channel: channel 1 (C1) will repeatedly broadcast segment 1 (S1), C2 will repeatedly broadcast S2 and S3, C3 will repeatedly broadcast S4 and S5, and so on. This technique uses a *control value* to restrict the maximum width allow in a segment, *W*, in order to not require a large caching space. There can be found two different loaders; an even loader, which download even-group segments, and an odd loader, which download odd-group segments. That way, a skyscraper receiver can simultaneously download from two channels; an even size fragment and an odd size fragment can start their download simultaneously. It can reduce the buffer space with 20%.

$$S_i = \begin{cases} S_1, & i = 1 \\ 2S_1, & i = 2,3 \\ 2S_{i-1} + 1, & i \bmod 4 = 0 \\ S_{i-1}, & i \bmod 4 = 1 \\ 2S_{i-1} + 2, & i \bmod 4 = 2 \\ S_{i-1}, & i \bmod 4 = 3 \end{cases} \quad (2)$$

- *Fast Broadcasting:* Fast Broadcasting divides each file into *n* equal segments and the server bandwidth is also divided into K equal channels. The difference is found in the distribution of the segments over the channels. This architecture uses the series *[1,2,4,8,16,32,64...];* channel 1 (C1) broadcasts segment 1 (S1) repeatedly, C2 broadcasts S2 and S3 repeatedly, C3 broadcasts S4, S5, S6 and S7 repeatedly, and so on. Hence, the number of segments must be $n = 2^k - 1$. Clients must receive data from all the channels and stored the segments downloaded but not yet required into their cache. Therefore, the client bandwidth requirement is equal to the server bandwidth and clients must store up to half of the video locally. The service delay is the *waiting time* for the first segment.
- *Pagoda Broadcasting:* Pagoda Broadcasting also divides each file into *n* equal segments and the server bandwidth into *K* equal channels. In order to distribute the segments, the technique uses the series *[1,3,5,15,25,75,125...]* but it does not require that the segments appearing on each channel be consecutive. Each channel is divided into time slots, each for a duration of a segment: slot0, slot1, slot2... Channel 1 broadcast repeatedly segment 1 (S1). Considering channel i there are two different cases:
    - i is even: Being *Sz* the earliest segment not broadcast in any of the previous channels, this segment is broadcast in every slot *multiple of z* (slot *jz (j=0,1...))*. The other even slots will be allocated to segments *Sz+1, Sz+2, ...,S3z/2-1*. The rest of the slots will be allocated to segments *S2z, S2z+2, ..., S3z-1*.
    - i is odd: Being Sz the first segment broadcast on channel i-1, the earliest segment not yet broadcast is *S3z/2*. This segment is broadcast in every slot *multiple of 3z/2* (slot *j3z/2 (j=0,1...))*. The following first slots will be allocated to segments *S3z* to *S2z-1* and the following slots of the

second repetition of *S3z/2* will be allocated to segments *S3z* to *S5z-1*. This pattern will be repeated as many times as necessary.

For a better understanding, see Fig. 8 Similar to Fast Broadcasting, the client must to tune in all channels to download data and store the segments downloaded but not yet required into their cache. The service delay is the waiting time for the first segment.

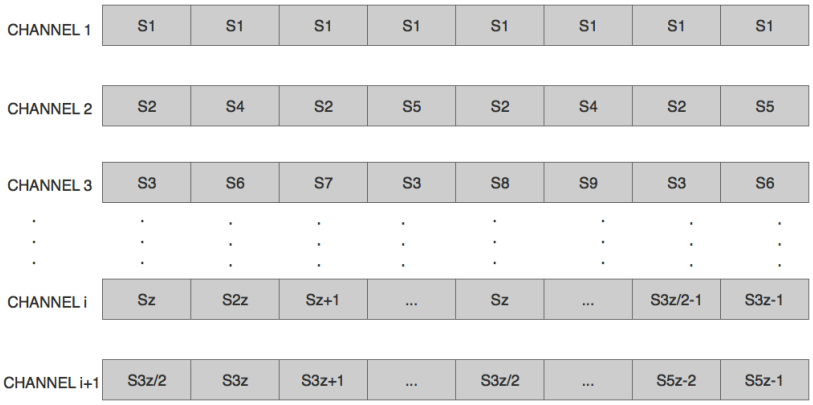

Fig. 8. Pagoda Broadcasting segments distribution

4. Caching and Prefetching

The popularity of Internet can threat its own utility, too many users accessing to the same Web Page produces an important waiting time for downloading contents and consequently a frustration in these users. However, many users access to the same information repeatedly, hence, there will be found redundant network traffic transmitted over the Internet. This situation and the user's experience can be improved using caching and prefetching techniques.

First, a definition of web caching is presented. The different caching architectures or types of caches, which can be useful for different applications, are found in section 4.2. Section 4.3 describes how these caches can be arranged and different replacement algorithms that allow the caches to have new and fresh data are presented in section 4.4. Finally, a definition of prefetching is presented.

4.1. What is web caching

The aims of web caching are to reduce the transmission of redundant network traffic and to improve access to the Web. In order to do that, web objects are temporarily stored closer to the end user for later retrieval.

There are notable advantages to Web caching, such as:

- Reduced bandwidth consumption: The number of requests and responses that have to be transmitted over the network is reduced. It benefits the user, the service provider and the website owner.
- Reduced latency: Web caching improves response times since the cache responses immediately and the content is closer to the user, which results in a faster delivery of Web objects to the end user.
- Reduced server load: Caching means that the server has to handle fewer requests.

But there are also some drawbacks of web caching if it is not performed correctly. The Web content in Web caches has to be regularly updated in order to not return stale content to the users and if there occurs a cache miss, that is a requested content is not found, the response time is increased. For that reason, a poor configuration of a caching system may degrade performance.

4.2. Caching architectures

There are several types of caches, some of them are:

- *Proxy Caching/Forward proxy caching:* The cache intercepts HTTP requests from clients and, if the object asked for is found in the cache, the cache gives the object to the user. Instead, if the requested object is not found in the cache, the proxy cache sends a request to the Web server hosting the original content. Once the object is fetched, a copy of it is stored in the cache before forwarding it to the user. The proxy cache is placed close to the user and normally it is found at the edge of a network, so a large number of users may be serviced. The cache hides the identities of clients.
- *Reverse proxy caching:* While a forward proxy is an intermediary in the side of clients, a reverse proxy is an intermediary in the side of servers. A reverse proxy cache accepts requests from clients on behalf of servers stationed behind it. It is helpful for servers that anticipate a considerable number of requests, in order to maintain its quality of service. It can also help the protection of the server; since a reverse proxy hides the identities of servers, a hacker would find it considerably more difficult to attack or acquire data found in them. Reverse proxy caches can also act as load balancers, distributing requests to a cluster of servers.
- *Adaptive web caching:* An important problem in Web caching is the "hot spot" phenomenon, where some Internet content become very popular and in high demand overnight, although after a short period of time they may not attract any interest. Adaptive web caching tries to optimize global data dissemination. It consists of several distributed caches that dynamically join cache groups (known as *cache meshes*) or leave such groups based on content demand. The adaptability and self-organizing property of meshes try to solve the cases in which demand for Web objects gradually grows or spikes or is otherwise erratic, unpredictable high or low. Adaptive caching uses two protocols: Cache Group Management Protocol (CGMP) and the Content Routing Protocol (CRP).
    - *CGMP:* This protocol describes how cache meshes are created and how a particular cache joins and leaves those meshes. In order to determine the admission or rejection of members from a group, caches use mechanisms such as feedback and voting. Caches, in general, are organized into overlapping multicast groups.

    - *CRP:* This protocol is used to locate cached content inside the set of meshes. It takes advantage of the overlapping structure of meshes for distributing object queries between groups and for propagating popular web objects throughout the mesh. This technique depends on multicast communication between members of the cache group and it uses URL tables to determine to which overlapping meshes the different requests should be transmitted.

- *Active caching:* This type of caching is trying to solve the problem of dynamic data on the Web. Dynamic data can cause problems when only conventional caching strategies are used. This type of data is growing in popularity, as Web users tend to prefer personalized services. Servers move part of their work to a proxy when requests come from a user, and active caching uses Java applets, located in the cache, that enable customization of Web objects. When a request for personalized content comes for the first time, the server provides the required Web object and any associated cache applets. When future requests are made for the same content, the cache applets perform the required processing at the cache that would otherwise be performed at the origin server, in consequence there is no additional expense of contacting the origin server.

An object in a cache can be in three different states 'fresh' objects, these are objects ready to be served as they are on date, 'stale' objects, these are objects that have to be re-cached as they have passed their expiration date or nonexistent objects, these are objects that have not been cached in the proxy cache.

4.3. Design techniques

Some design techniques used to arrange caches are:

- *Hierarchical caching:* Hierarchical caching arranges a series of caches in a tree-like construction. Caches are classified into parent caches and child caches, and usually, child caches can query parent caches and each other, but parents never query children. When a request arrives, if a cache can't response to it, the request will progressively go down the hierarchy until a cache can provide the requested object. This architecture can cause an overload in parent nodes due to many queries of their children. One possible solution is to employ clustering.

- *Distributed caching:* In distributed caching all the caches in the upper level are used to possess information about the content of the leaf caches, the only participant caches that fetch data. The upper level server returns a 'pointer' to the requested objects consisting of a URL and information about the size and modification time of the object. This architecture allows a rapid location of requested objects and disk space usage is also reduced compare to hierarchical caching, due to upper level nodes only need to store pointers to objects and not copies of them.
- *Hybrid caching:* In hybrid caching architectures, caches of the same level work together by employing distributed caching.

4.4. Cache replacement algorithms

Cache replacement refers to the process that take place when old objects are removed to incorporate new ones.

The most important factors that can influence the replacement of an object are:

- *Recency:* time since the last reference to the object.
- *Frequency:* number of requests/retrievals to an object.
- *Cost of fetching*
- *Size*
- *Modification time:* time since the last modification.
- *Expiration time:* time when an object has to be immediately replaced.

There are several ways to categorize cache replacement algorithms, but in this thesis we will use the classification suggested by Aggarwal et al. [1999]:

- *Traditional Algorithms:* These replacement algorithms include:
    - *Last Recently Used (LRU):* This cache algorithm places the most recent objects near the top of the cache. When a new object is accessed, it is placed at the top of the cache. When the cache limit has been reached, it removes the least recently referenced objects that are placed at the bottom of the cache.
    - *Last Frequently Used (LFU):* This method removes the last frequently requested object.

- *Pitkow/Recker:* The base of this algorithm is LRU. It changes its performance when objects are requested on the same day; they will be differentiated by their size and the biggest ones are removed first.

- *Key based Algorithms:* These strategies sort objects upon a primary key.
  - *SIZE:* This algorithm removes the object with the largest size. When objects have the same size, it is applied the LRU strategy.
  - *LRU-Min*: This algorithm aims to favor smaller objects. If there are objects in the cache with size at least S, the algorithm removes these objects in LRU order. In case there are no objects with size at least S, the threshold is set to S/2, (objects with size at least S/2 are removed in LRU order).
  - *Hyper*-G: It combines LRU, LFU and SIZE. First, objects are chosen in LFU order. Then, ties are broken by choosing the least recently used (LRU order). If there is still not a unique object, the largest object is chosen.
  - *Lowest Latency First (LLF)*: This strategy tries to improve the latency incurred by requesting an object from its original server. It tries to keep the cache objects that originate from slow serves in order to reduce the latency experienced. For that reason, LLF removes the fastest-to-get objects.
  - *Hybrid*: An hybrid algorithm computes a function for every object (i) that takes into account the time to connect to the original server (C_s), the available bandwidth to the server (b_s), the number of times the object has been requested and its size. The function is shown in (3). W_b and W_n are weighting parameters. The values for the time to connect to the original server and the available bandwidth to the server, are based on the values that were calculated in the past when the proxy contacted to the server s.

$$\mathrm{f(i)} = \frac{(C_s + \frac{W_b}{b_s}) f_i^{W_n}}{s_i} \tag{3}$$

- *Function-based algorithms:* These algorithms use functions for sorting objects, using parameters such as time since the last access was made, size of the object, the time an object has been in the cache, the time of expiration of an object etc. There can be found different algorithms and different functions, one for each. Some examples are: Greedy-Dual-Size (GDS), Lowest Relative Value (LRV), Least Unified-Value (LUV) etc. As they are not important for understanding the thesis, and the basic algorithms have been already explained, there won't be a further explanation about function-based algorithms.

4.5. What is prefetching

The aim of prefetching is to fetch information from servers before being requested. An accurate performance of prefetching can notably improve the quality of service of users. Access latency can be reduced, and if data can be prefetched accurately and early enough, memory won't have to access the server and therefore the latency produced by this access will be reduced. Prefetching can also reduce cache misses (when an object is not found in the memory). However, prefetching useless data can waste resources such as memory bandwidth, buffer space or energy consumption.

In order to know what data has to be prefetched, a prediction engine is used to predict, based on previous access patterns, the object to prefetch. It is also important to decide the exact time to do the prefetch request; if it is too early, prefetched data might not be used and if it is too late, the memory latency won't be reduced.

5. Network traffic analysis

The use of network measurements can improve the performance of every network. Collecting information about Internet traffic helps to understand the behaviour of a specific network, and this knowledge can be applied to a variety of tasks like workload characterization, work troubleshooting, control policies and performance evaluation and improvement.

This section explains the measurement methodology used, how it is characterized and the differences between the chosen features. Finally the concept of Internet traffic models is introduced and some of the models that had been/are being used.

5.1. Measurement methodology

There exist several different ways of measuring the traffic in a network. To characterize the measurement approach used to study a concrete network, the following features can be defined:

5.1.1. Hardware-based or Software-based measurement tools

A hardware-based tool is often known as network traffic analyzer, as its name implies, its purpose is to collect and analyze network data. Although a hardware-based tool usually offers better performance than a software-based tool, it is more expensive and its cost depends on features such as number of network interfaces, types of network cards, storage capacity and protocol analysis capabilities.

A software-based tool typically modifies the kernel-level of commodity workstations' network interfaces in order to provide them with packet capture capability. With a software-based tool, it is possible to capture TCP/IP packets and analyze features such as the source port, the destination port, the time the packet was transmitted etc. This kinds of tools are less expensive than the hardware-based tools, but the performance of a dedicated network traffic analyzer may be more accurate.

5.1.2. Passive or Active measurement approaches

A passive network monitor is a non-intrusive measurement device; which observes and records packet traffic on a network without injecting additional traffic.

An active network monitor uses packets generated by the measurement device to probe the Internet and measure its characteristics.

5.1.3. Online or offline analysis

An online analysis is available in network traffic analyzers that support real-time data collection and analysis. These analyzers often provide graphical displays of live traffic data.

Other network measurement devices only support real-time collection and storage of traffic data. For that reason, it is necessary to first collect and store the data in order to perform then the analysis of it, but always in a post-processing phase.

5.2. Internet Traffic Models

In order to plan the capacity of networks, it is necessary to predict how the network will act depending on the characteristics of the actual traffic. Control policies, which allow an optimal utilization of a network's capacity, are chosen depending on the performance modelling of every network, and at the same time, these performance models require accurate traffic models to capture the statistical characteristics of the traffic on the network.

There are different models, which are used depending on the complexity and precision of the study.

5.2.1. Renewal Models

In renewal models the arrivals are independent and identically distributed, but, in contradiction to this independence assumption, Internet traffic is often highly correlated. For that reason, an assumption of an uncorrelated traffic might result in unrealistic models.

- *Poisson Distribution Processes*: Poisson models are the oldest traffic models and one of the most used. They make two primary assumptions: the number of sources is infinite and the traffic arrival patterns are random. It is a memoryless model, which makes it very attractive from an analytical point of view due to the simplification of queuing problems involving Poisson arrivals. Another interesting

property is the fact that the aggregation of multiple Poisson streams generates a new Poisson stream whose rate is the sum of the component rates.
Interarrival times of a Poisson process (A_n) have an exponential distribution with rate parameter λ:

$$P(A_n \leq t) = 1 - e^{-\lambda t} \quad (4)$$

Equivalently, the probability of *K* arrivals in an interval is defined as

$$P(X(t) = k) = \frac{(\lambda t)^k}{k!} e^{-\lambda t} \quad (5)$$

Studies such as [10] prove that Poisson model is not suitable to describe Internet traffic in modern Local Area Netoworks (LAN) and Wide Area Networks (WANs), due to the importance in this scenarios of batch arrivals, event correlations and the bursty behaviour of real traffic, while in Poisson processes all arrivals are considered independent and identically distributed.

- *Bernoulli Processes*: Bernoulli processes are the discrete-time analog of Poisson processes. The probability of an arrival in a time slot *k*, is *p*. The number of arrivals for a given slot, *k*, is a binomial variable:

$$P(N_k = n) = \binom{k}{n} p^n (1-p)^{k-n} \quad (6)$$

5.2.2. Markov Models

Markov models introduce dependencies between requests. It models the activity of the traffic source on a network, by a finite number of states. The probability of the next state depends only on the current state. The set of random variables referring to different states {*Xn*} is called a Discrete Markov Chain. In the discrete-time case, the state transitions happen only at integral values and the time spent in a state is geometrically distributed. Given a continuous-time Markov chain, the state changes can occur at any time *t* and the time spent in a particular state is exponentially distributed.

- *Markov Modulated Traffic Models:* Markov modulated traffic models introduce an explicit notion of state into the description of a traffic steam, that is, the current state modulates the probability law of the traffic mechanism. If *M* is a Markov process and *M* is in state

k, a probability law for arrivals takes effect for the duration of this state *k*. When *M* changes the state to *n*, a new probability law of traffic arrivals takes effect while *M* is in *n*.

- *Markov Modulated Poisson Processes (MMPP):* A Markov modulated Poisson process is a Markov modulated traffic model, which combines Markov processes and Poisson processes. In this scheme, in state *k* of a Markov chain (*M*), arrivals occur according to a Poisson process with rate λ_k. Rate changes as state changes.

5.2.3. Linear Stochastic Models

In contrast to Markov models in which the next state depends only on the current state, autoregressive models define the next random variable in a sequence as an explicit function of previous ones within a time window stretching from the present into the past. These models are suitable for modelling short-range dependencies.

- *AR(p):* AR(p) is the simplest form of a linear autoregression scheme, where *p* is the order of the autoregression. Its definition is the following:

$$X_n = a_o + \sum_{r=1}^{p} (\alpha_r X_{n-r}) + \varepsilon_n, \quad n > 0 \tag{7}$$

Where *X* are random variables, α_r are real constants and ε are zero-mean, independent and identically distribution random variables (known as residuals or innovations), which are independent of *X*.

5.2.4. Self-Similar Traffic Models

In the early 1990s, [11] showed evidence of self-similarity in network traffic. Hence, previous models were shown to be incomplete or invalid.

A self-similar process behaves the same at all scales, for instance a self-similar network trace would look the same aggregated in 1second bins as aggregated in 10 second bins. This behaviour is not found in the case of Poisson traffic, an increase of bin sizes will smooth Poisson traffic and it eventually will reach a flat line at the distribution mean.

Self-similar process can be described by heavy-tailed distributions. An example is the Pareto Distribution Process. It produces independent and

identically distributed interarrival times. The probability that *X*, a random variable with Pareto distribution, is greater than *x* is

$$P(X > x) = \left(\frac{x}{x_m}\right)^{-k} \; for\ all\ x \geq x_m \tag{8}$$

Where *k* is a positive parameter.

The probability density function is given by:

$$p(x) = \frac{\alpha k^{\alpha}}{x^{\alpha+1}} \; , k \leq x, \quad \alpha, k > 0 \tag{9}$$

Where α represents the shape parameter and *k* represents the local parameter of distribution.

Pareto is good for modelling situations where items are highly skewed.

PART II: Related work.

6. Previous work on traffic measurements

An summary of some previous studies of Internet traffic is found in this part of the thesis. In section 6.1, we present information about the evolution of traffic over the Internet and also specifically about multimedia streaming services. In section 6.2 the user behavior was characterized. As VOD systems are one of the most popular applications, in section 6.3 we present information about prefetching and catching in that type of systems and the results that were found.

6.1. Traffic analysis

In [1] a 5 year long traffic investigation was reported, from June 2007 to May 2011. This study showed an annual growth rate of the average daily total traffic of 6%. Even though the P2P file-sharing remained the dominant network bandwidth consumer over the five years mentioned, the daily file-sharing traffic per end user largely remained the same and already reached its peak at the beginning of the studied period of time and even suffered a major drop from the first half year of 2009. This drop was attributed to the Swedish enforcement of the European Union's Intellectual Property Rights Enforcement Directive (IPRED) effective from April 1. Hence, we can see that changes in the background of the actual year can force a change in the user behavior.

Although much of the traffic volume has been attributed to file sharing traffic, the daily streaming media traffic volume per end user has experienced an annual growth rate of about 40%. If we talk about the differences between the inbound and outbound traffic, the study showed that, focusing on file sharing traffic, the outbound traffic share has been significantly more than the share of the inbound traffic, and the reason is the popularity of P2P applications.

But, streaming media has emerged to be the second largest contributor, from amounting to only 2% of the total network traffic in 2007 to up to 13% of the total network traffic in the first 5 months of 2011. For streaming media the downlink inbound traffic had a dominantly larger share, amounting up to 26% of the total inbound traffic in the first 5 months of 2011.

There are publications focused more specifically on multimedia streaming services. In [2] the study analyses the flows for different multimedia services from the 25th of March 2011 to the 31st of March 2011. It was found that the streaming services made up 24% of the

incoming bytes and 9% of the incoming connections. What is more, we can see that a Swedish P2P based video application, Voddler, is the dominant one, generating 78.9 million connections during the measurement period of one week. This lead to a change in the flow patterns of streaming media; in general the inbound traffic was significantly larger than the outbound traffic, as we have seen before, but in the case of multimedia streaming services such as Voddler, the opposite applies.

The same analysis found another dominating streaming service characterised as a P2P application, Spotify, however, in 2014 Spotify moved away from using P2P networks. Therefore, there might be found some changes in the analyses' results if they are done nowadays.

A conclusion taking into account both studies would be that streaming-based new video and audio services such as Youtube, on-line TV etc. are the driving forces for the increase of the total network traffic generated by the end users. These reports were performed 5 years from now, and the growing popularity of applications such as Netflix might have changed the Internet traffic, increasing the daily streaming-media traffic volume; hence the file sharing traffic might not be the overriding traffic actually.

6.2. User behavior

Another important part for understanding how the Internet traffic is generated, is the study of the users' demand patterns. In contrast to the analysis of Internet traffic, users' demand patterns aren't likely to change over the years, as we will see in future sections. For that reason, we can use the results of [3], from 2009, as if they would have been produced nowadays.

The study shows that there are many active IP addresses during the evening; with a peak between 20:00 and 22:00. The activity decreases from 00:00, and it reaches the minimum value at 6:00 and then starts to increase again. If we focus on the traffic rate, more than 90% of the data have a traffic rate less than 2,5Mbps.

Also, the paper analysed some characteristics of active sessions. There were mainly short sessions, but some of the sessions lasted the whole measurement period, two weeks, which is due to the fact that some users have P2P file sharing applications running constantly.

In [2] the users' demand patterns of streaming media services are presented. As in [3], the flow statistics follow a periodic daily pattern, with peaks in the evening around 21:00. Focusing on the bytes, there are several services that generate a huge amount of flows, but not much data. We can

relate it with the results found in [3] where 90% of the data have a traffic rate less than 2,5Mbps.

Since the streaming media traffic tends to increase and Video-on-demand (VOD) services are the mainly reason for that raise, we are going to focus on users' behaviour concerning VOD.

6.2.1. VOD users behavior

From [4] we can see that the hour with least VOD request arrivals is at 5:00 and the hours with the highest arrival rates are from 20:00 to 22:00, more or less the same behaviour as explained before.

More than 50% of the sessions start after one or less than one second from the arrival of the previous sessions and around 90% of the sessions start within a minute from the previous session.

One important aspect of VOD is the length of the sessions. In the paper, the concept of "impatient user behavior" is discussed, where users abandon streaming sessions after a short period of time of starting them. The numbers found were that more than 90% of the sessions last for less than one hour and more than 50% last less than 12 minutes. If we compare these results with the ones in [5] that are dated 2004, the patterns are similar.

An interesting behaviour if we consider the popularity of videos is found in the length of the sessions. Videos with a medium popularity have the longer session times. The conclusion in [5], where the same results were found, is that people who is watching the most popular videos are likely to have seem them before and therefore popular videos suffer from loss of interest after repeated viewings.

In [4] we can see that the less popular videos were streamed only once during the measurement period and that the median time for a session for these videos is less than 200 seconds. The median time between two streaming sessions decrease as the video popularity increases, as it is expected because more arrivals will occur.

6.3. VOD-Caching and prefetching

The use of caching and prefetching in VOD systems can lead to a better user experience. Studies have shown that user demand patterns are well characterized and hence they have good potential caching gains.

When using caching, the system first checks if the request content is found in the local cache. If the content does exist, then it is delivered to the user immediately. Otherwise, if the content is not in cache, then it is fetched

from a server located somewhere on the web. Then, depending on the size of the content and the transmission rate, the delivery of the content could take a significant amount of time.

Without a prefetching scheme, the content, in our case video/audio, will be stored in the local cache once the content has already been watched/listened.

Considering an analysis of user behaviour, the results can be used to build a prefetching scheme and reduce user perceived latency. The objective of the prefetching system is to proactively preload certain content to the cache even before the user requests it.

In [6] the performance of prefetching N episodes of a series was studied. This scenario is one of the most favourable, since if a user sees one episode the user is more likely to continue watching more episodes in a near future. Then, in order to see the power of prefetching, we will present an analysis of the results in [6].

6.3.1. Prefetch N episodes

The probabilities that a request for episode n will be followed by a request for episode n+k are the following: for a user who has watched episode n of a series; the probability of that user watching episode n+1 next is over 50% and the probability of that user not watching any episode after n is 26%.

Then, prefetching videos with index of: n+1, n+2, n+3, n-1,n+4, n+5 and n-2 will account for 95% of the requests for a next video.

In [6] we can also see the improvement in the cache hit ratio when prefetching different number of episodes. When no videos are prefetched, but passive caching is used, the hit ratio reaches 13.7%. However, just prefetching episode n+1, the hit ratio can reach 55% and if episode n+2 is prefetched the hit ratio becomes 60%. Prefetching further videos after n+1 and n+2 gives very little increase in the hit ratio.

6.3.2. Length of prefetched videos

Taking into account the behaviour exposed in section 5.3.1 and also the results found in [4] the favourable length of prefetched videos can be determined.

Since the longer session times are found in videos with a medium popularity, the following can be determined:

- Caching the first 16 to 20 minutes for videos with medium popularity.
- Streaming the first 3 minutes for videos with low popularity.
- Caching and prefetching the first 10 minutes for videos with high popularity.

6.3.3. Time to prefetch

It is important to know when the prefetching engine has to upload the videos to the local cache in order to not cause a major traffic overhead and also to have the videos available before the user demands them.

One profitable time to prefetch as much content as possible is during off-peaks hours; from 02:00 to 06:00, for example. Since the traffic load in those periods of time is low, the prefetched data won't cause any congestion to the network.

It is also interesting to know the estimated available time for prefetching between two consecutive viewing sessions. This study is also found in [6]:

In it, we can distinguish between the sequential views, the behaviour of watching n+1 or non-sequential views, and the behaviour of watching any video not in sequential order. The 30% of the sequential views arrive within 20 minutes while only 15% of the non-sequential views requests are generated within 20 minutes. If we take a closer look, 40% of the sequential views are generated within 1 minute after the end of the viewing session and only 15% of the non-sequential views are generated within the same time period. Therefore, if videos are prefetched at the end of the current sessions, the time frame is limited. Thus, it is reasonable to prefetch the next video in order during the current session.

Prefetching for sequential views means prefetching one episode only beforehand, whereas prefetching for non-sequential views means prefetching more episodes after the next one in order. For non-sequential views, the request pattern shows that people watch episodes from the same program on daily basis, which means a longer time period for prefetching that can be delayed until off-peak time.

PART III: Renewed data analysis.

7. Evolution of Internet traffic from 2007 to 2015

Taking into account the results found in [1], where an analysis of 5 years of Internet traffic was presented, and the new data collected from 2013 to 2015, we have performed an evolution analysis focusing on Internet traffic.

In section 7.1 and 7.2, we present a description about the target network and how the data has been collected. Once the target network has been characterized, the Internet traffic is grouped into the same application categories as in [1] in order to compare the results and do a proper evolution analysis. In section 7.3 the shares' evolution of every category from 2007 to 2015 are presented. Later, in section 7.4 the user behavior pattern of 2016 is analysed and compared with previous years. Finally, section 7.5 shows the evolution of the most popular video games.

7.1. Target network

The analysis in this thesis is based on data from one network denoted Network North. This network as a fiber-based, open network with approximately 7000-8000 customers connected, most of them residential, in about 2500 households. The measurements are performed by the network operator and the anonymized data is made available to Acreo for post-processing.

7.2. Data collection

The traffic data collection tool used was PacketLogic (PL), a commercial traffic management device. In PL, traffic is identified based on packet content (deep packet inspection and deep flow inspection) instead of port definitions. The PL's Filtering module uses PL's Datastream Recognition Definition Language (DRDL) [9] to identify which application protocol is generating each connection. The extracted traffic information allows the definition of variables such as the inbound and outbound traffic, the user name, the file name or Web URL etc. The identification process is connection-oriented, which means that each established connection between two hosts is matched to a certain application protocol. It is during this identification process that DRDL extracts the traffic properties. There are hundreds of services (HTTP, SMTP, FTP, Direct Connect, SSL, SSH...) as DRDL signatures and the signature database is updated constantly.

PL uses the traffic in both directions in the identification process and records all the traffic that pass through it in the form of the traffic volume, the traffic application, the actual timing, and the IP address for each traffic records during the 5 minute period of time.

For collecting the traffic data, the PL was connected to the network via optical 50/50 splitters. The measurement point is the Internet Edge (IE) aggregation point, where the service providers are connected to the network [7]. In order to enable long-term study of the target network traffic statistics, a MySQL data base was established to store the original traffic records collected by PL. When transferring data to the MySQL server, the IP addresses are hashed, ensuring that no violation of integrity or law is done. In parallel, the log data of the DHCP server of the network are also stored in the same data base. The DHCP server log contains information as timing, (hashed) IP address, broadband service subscription type, access switch and access port. Thus, in linking the two data set tables (PL record data and DHCP log) in the same data base, each traffic record collected by PL can be traced to e.g. the end user's broadband subscription type, the access switch etc., and by matching the (hashed) IP addresses of the two data sets, the traffic that were not generated by private end users can be excluded in the data analyses, as is the case in this study.

7.3. Category shares' evolution

Traffic is grouped into different application categories as in [1].

- *File Sharing:* File Sharing includes peer-to-peer (P2P) file sharing traffic and client-server file sharing traffic. This group doesn't include HTTP-based direct download file sharing traffic offered by e.g. RapidShare and Megaupload.
- *Streaming Media:* Streaming media includes both video and audio streaming traffic, such as flash video over HTTP (e.g Youtube), HTTP media stream, real-time streaming protocol (RTSP), and real-time messaging protocol (RTMP) family. P2P media stream is also included in this category.
- *Web Browsing:* Web Browsing is traffic generated by HTTP including its plugis. Note that the traffic generated by file hosting direct download services (RapidShare, Megaupload etc.) as mentioned above is classified into this category (as HTTP download traffic).
- *Others:* Other traffic includes traffic categories of *File Transfer, Messaging and Collaboration, Entertainment, Network*

Infrastructure, Business Systems, Remote Access, Information, and *Malware.*

The data used was extracted from January 2013 to December 2015, although there are a few discontinuity periods of traffic data collection during the studied period of time.

Fig. 9 shows the evolution of the input traffic of the different application categories. Fig. 10 shows the evolution of the output traffic of the different application categories. And finally, Fig. 11 shows the evolution of the total traffic of the different application categories.

As can be seen, file sharing traffic has been dominating the inbound traffic until 2013. If we assume a lineal progression from 2011 to 2013 due to the lack of data between these years, the exact point of this behaviour change would be the middle of 2012. From this point, streaming media inbound traffic is the dominant one. Fig.12 shows the streaming media contributors' shares from year 2013 to 2015. These contributors are Peer-to-Peer applications, Netflix, Spotify and Others. In Others we can find the traffic from services such as Vine, Abacast, Octoshape, etc and their contribution is less than 1% to the total streaming traffic. As noticed, the service, which contributes more to the inbound streaming media traffic, is Netflix; Netflix inbound traffic increases from a 30.8% in 2013 to a 38.61% in 2015. For that, it can be conclude that the most popular service considering streaming media is Netflix. Considering the total traffic evolution, it can be noticed that from the middle of 2014 streaming media traffic is the dominant total traffic. This is due to the increase of the streaming media inbound traffic (since outbound traffic changes are not that sharped); streaming media inbound traffic has increased from a 26% in 2011 to a 42% in 2015, while file sharing inbound traffic has decreased from a 87% in 2011 to a 70% in 2015. Moreover, the inbound traffic share in streaming media is significantly more than the outbound traffic share for the 8 years.

If we focus on the outbound traffic, file sharing traffic is still by far the dominant one, but its popularity is decreasing. If we take a closer look to the contributors of this category, we can find two types of applications: Client-Server and Peer-to-Peer applications. The most popular services belong to Peer-to-Peer, which explains the fact that the outbound traffic in file sharing is higher than the inbound traffic. Some examples of Peer-to-Peer applications would be; Ares, BitTorrent, eDonkey... Due to its popularity, the behaviour of file sharing traffic is related to the behaviour of Peer-to-Peer applications. Its inbound traffic shares have decreased from a 26.9946% in 2013 to a 13.9986% in 2015 and its outbound traffic shares

from a 77.9844% in 2013 to a 69.993% in 2015. As can be seen, the decrease is more significant in the inbound traffic. This can explain a change in the users' behavior; the fact that the outbound traffic hasn't decreased as much as the inbound traffic suggests that users are still using these applications, but they are not downloading as much data as before although they are still sending data to other peers.

To be able to see better how the inbound and outbound flows have been changing due to the changes in the different category shares, Fig. 13 shows the amount of traffic of March from 2013 to 2015. As can be seen, inbound traffic has increased considerably, from 1703200Gb to 227200Gb, on the other hand outbound traffic has decreased from 1588400Gb to 53479Gb. The increase of the inbound traffic and the decrease of the outbound traffic reinforce the idea that streaming media has been gaining popularity over file sharing applications.

An interesting fact is the increase in the category Other. Considering the shares of every application separately, as shown in Table I , we can see an increase of 10% in the network infrastructure total traffic that has produced the increase shown in Fig. 11.

Additionally, web browsing outbound traffic share remains largely the same at the level between 1-3% although it can be noticed some fluctuations in the inbound traffic share; from 2011 to 2013 shares increase from 12% to 17% while in 2015 it has decreased to over 10%.

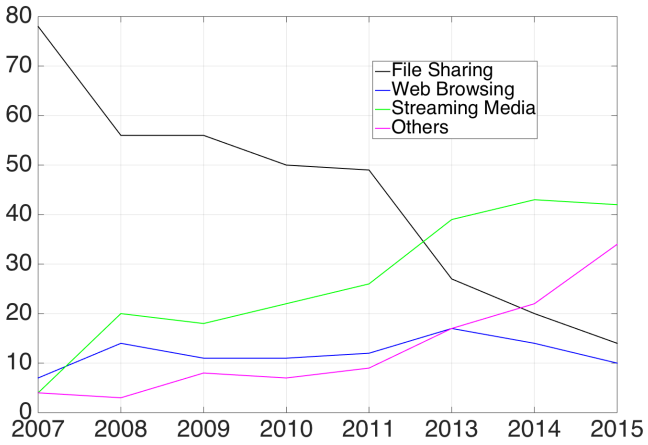

Fig. 9. Inbound traffic evolution

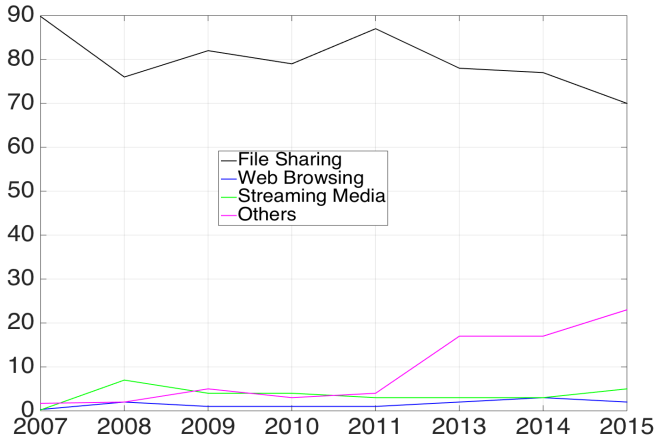

Fig. 10. Outbound traffic evolution

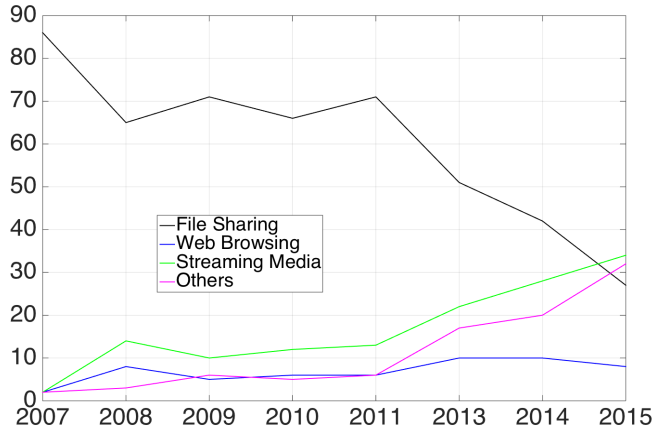

Fig. 11. Total traffic evolution

TABLE I. OTHERS APPLICATIONS' TOTAL TRAFFIC SHARES

Application	2013	2014	2015
Business System	<1	<1	<1
Entertainment	<1	<1	<1
File Transfer	2	3	5
Malware	<1	<1	<1

Messaging and Collaboration	1	2	2
Network Infrastructure	12	13	22
Remote Access	1	1	1
Information	<1	<1	<1

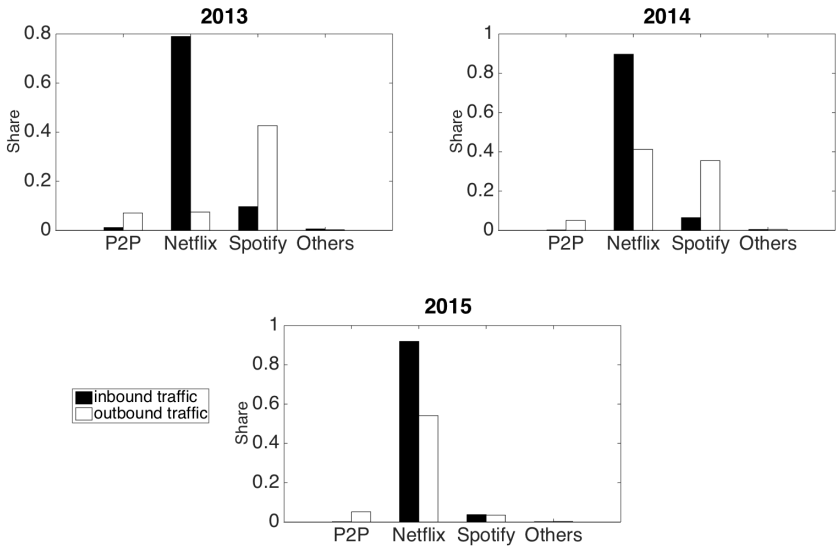

Fig. 12. Streaming media contributors' shares 2013-2015

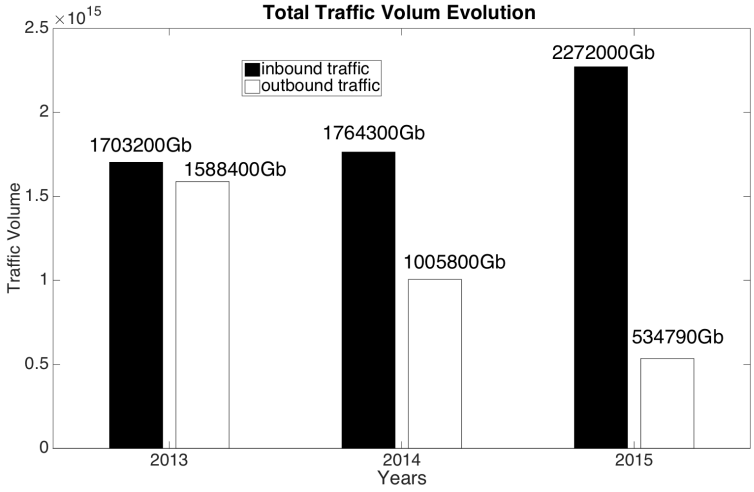

Fig. 13. Total traffic volume evolution 2013-2015

7.4. Users' behavior 2016

The users' behaviour of 2016 has been analyzed in this section. Two different analysis can be found; firstly, we have observed the daily Internet traffic pattern, that is how the Internet traffic is produced depending on the time of the day, and finally, the length of the users' Netflix sessions.

7.4.1. Daily Internet traffic pattern

Also, data has been collected from January 2016, in order to compare the users' access patterns with the ones found in previous years.

Fig. 14 shows the daily Internet traffic pattern for the network. The traffic pattern for weekdays and weekends are very similar, for that reason only the pattern for a weekday is shown. As can be seen, Internet traffic decrease from the beginning of the day until 7:00, when the traffic starts to increase again, and this behavior can be related to the beginning of the workday. There are also peaks from 20:00 to 22:00. If we compare the results found in 2016 and the ones in [3] dated from 2009 and explained in section 6.3, we cannot find any relevant changes, hence users' access patterns haven't changed over the years.

If we focus the analysis on the asymmetric behavior of the network, we can find an important change. In [3] Internet traffic is also asymmetrical, but with more outbound traffic than inbound traffic. If we look Fig. 14 it is

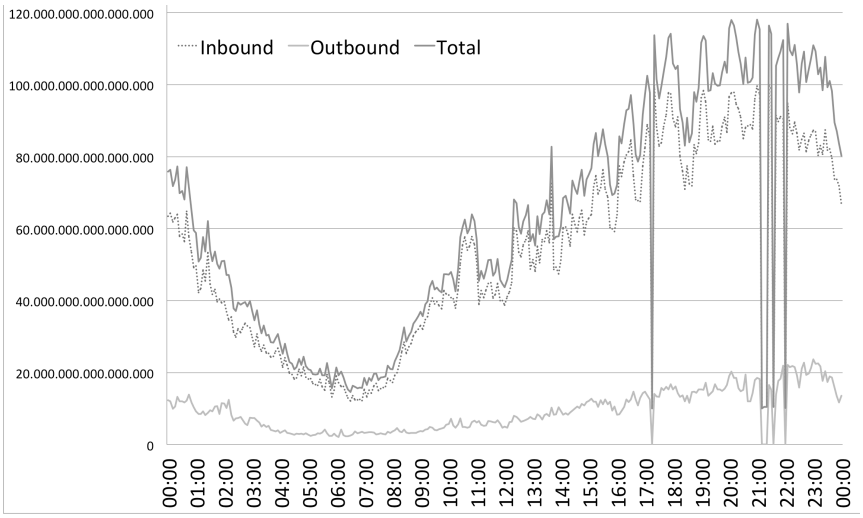

Fig. 14. Daily Internet traffic pattern

shown the opposite, there is more inbound traffic than outbound traffic. This change is related to the changes found in the different applications' shares: as was exposed in section 7.3, file sharing traffic has become the second total contributor, while in previous years file sharing has always been the major contributor, far from others categories. It was also shown that the most important file sharing services are P2P and these services produce more outbound traffic than inbound traffic. Streaming media, on the other hand, become the major contributor, producing more inbound traffic than outbound traffic. The decrease in popularity of file sharing and the increase in popularity of streaming media have produced a change in the symmetrical behavior of a network.

Fig. 15 shows the total traffic for different days of a week. As expected, weekends have the major amount of Internet traffic. This result reinforces the concept that Internet usage is related to users' free time as a way of entertainment.

7.4.2. Length of Netflix sessions

As shown in section 7.4, Netflix is the main contributor to the Streaming Media traffic. We have used data from January and February 2015 in order to analyze the length of the users' Netflix sessions.

Fig 16 shows the cumulative distribution function (CDF) of the length of the Netflix sessions. Axis x is in units of minutes/5.

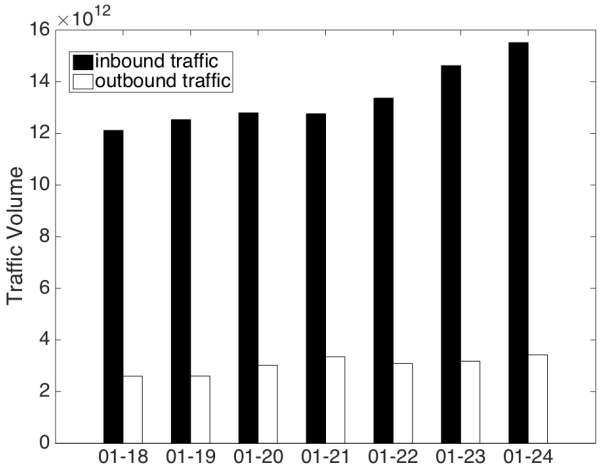

Fig. 15. Weekly Internet traffic pattern

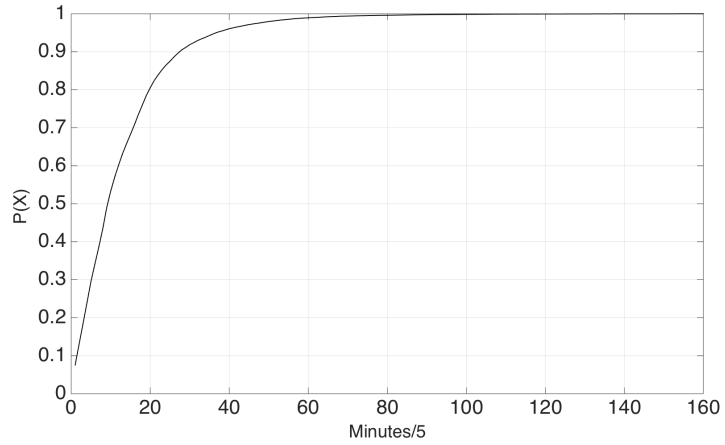

Fig. 16. CDF of the length of the Netflix sessions in minutes/5

All data corresponding to sessions of 0 minutes was omitted due to the impossibility of extract the exact seconds, hence these sessions can be considered as failed sessions. As can be seen, the majority of the sessions have durations of less than an hour, where 50% of them lasted less than 45 minutes and 20% lasted less than 15 minutes. 90% of the sessions lasted less than 135 minutes.

Fig 17 compares the probability distribution function (PDF) of the length of the Netflix sessions with an exponential distribution with mean value 13.3782 minutes/5, which corresponds to a mean session length of

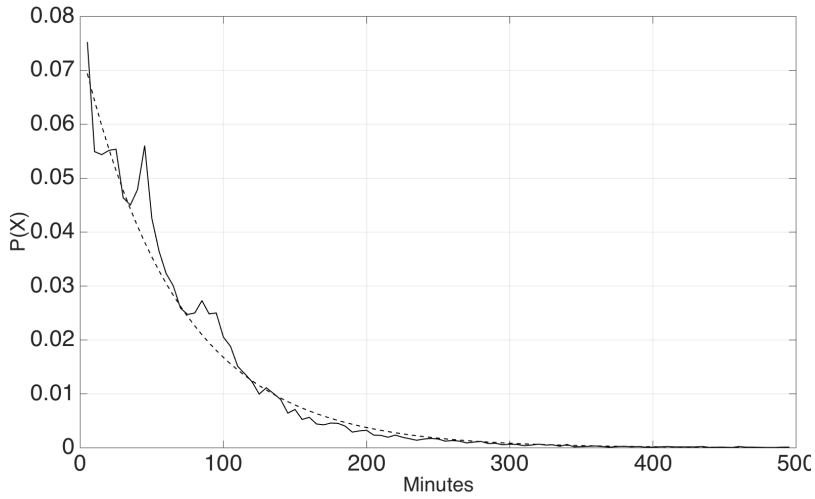

Fig. 17. PDF of the length of the Netflix sessions compared with an exponential distribution with mean value 13.3782

66.891 minutes, a bit more than an hour. Both pdfs fit well, although there are some peaks around 40 minutes length and around 90 minutes. The majority of the episodes of a series last between 40 to 45 minutes, which can explain the first peak in the pdf around this value. On the other hand, movies last between 90 to 120 minutes, which explains the other peak found in the pdf. If we compare the probability of both peaks, the most probable length is the one corresponding to the view of an episode of a series, which could imply that the majority of users use Netflix to see series instead of films.

7.5. Evolution of popular video games

The video games industry revenue is increasing year by year, which is why it would be interesting to make an analysis about the Internet traffic that they produce and the most popular games. For now, entertainment traffic, including gaming, is not one of the most important categories; as it is shown in table I the Entertainment traffic share is less than 1%, but which are the most popular games?

Fig. 18 shows the different shares of every game from 2013 to 2015.

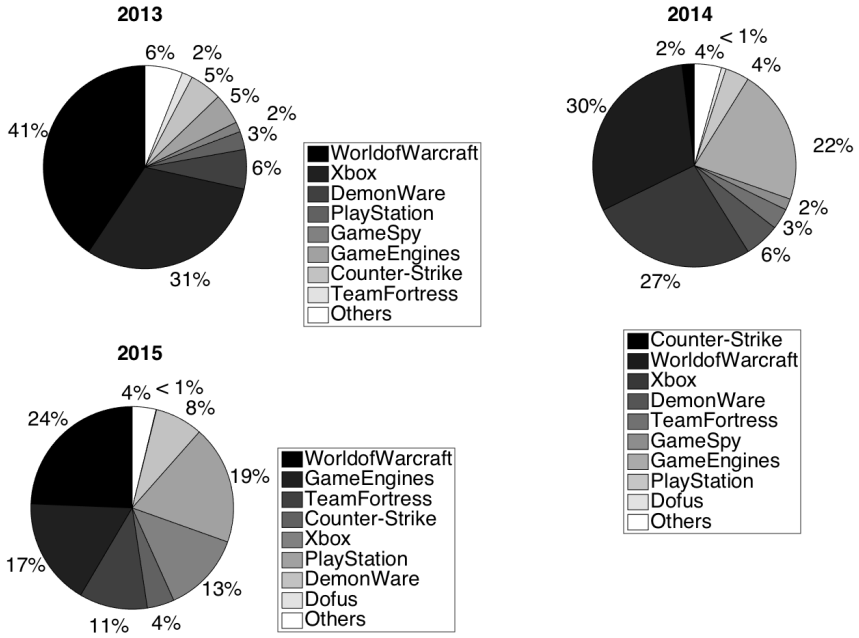

Fig. 18. Video game shares

As can be seen, World of Warcraft is the dominant game from 2013 to 2015, but its popularity is decreasing; from 41% of the total gaming traffic in 2013 to 24% in 2015. If we take a closer look at the inbound and outbound traffic of specifically this game, we find that the traffic is decreasing too; it is not only that other games are gaining much more popularity, it is that the number of users using World of Warcraft is decreasing. This behavior is shown in Fig. 19. World of Warcraft was created in 2004, thus it has suffered from many changes since then. The main critiques concerning the game are its current easiness due to a game modification and the impossibility to do interesting tasks once a high level is reached. These issues have caused a leak of users to other games (not necessarily of the same type), such as League of Legends, Runes of Magic, The Exiled Realm of Arborea, Guild Wars etc.

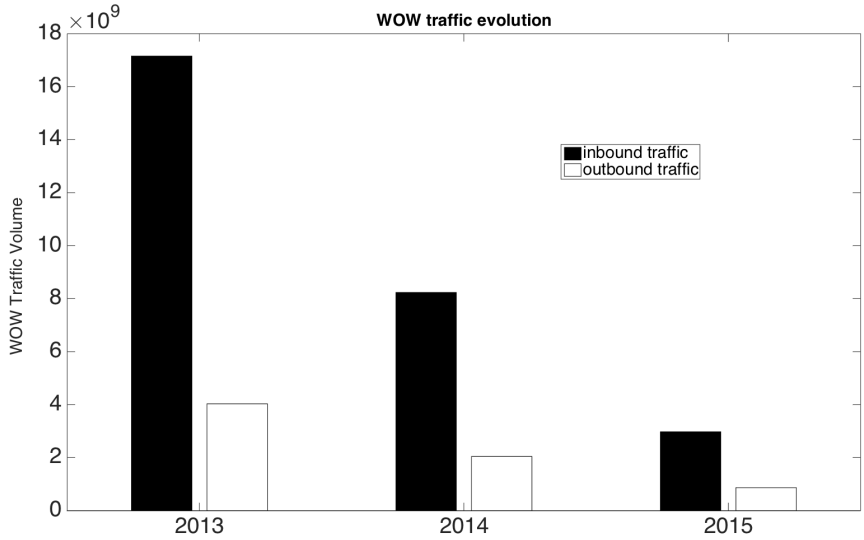

Fig. 19. World of Warcraft traffic from 2013 to 2015

8. Conclusion

The study performed in this thesis has proved that audio and video services, in concrete Netflix and other VOD systems, are the driving forces of the total network traffic. This assumption was concluded in study [1] and now it has been proved.

If we relate to the decrease of the inbound file sharing traffic with the growing popularity of VOD systems, we can conclude that users now have other ways to see their series or movies, and they are no longer using file sharing to download this data. Instead of downloading series or movies, they prefer to use services that provide videos in high quality despite of their cost.

This study has also shown that users' access patterns are still linked to the daily life of users. For that reason, there aren't any relevant changes in the results found from 2016. The majority of data is collected in the afternoon, when the workday has finished and users are at home. But, while access patterns are the same, the asymmetrical behavior of the network has turned to be the opposite; with more inbound traffic than outbound traffic, which will allow the traditional implementation of broadband access network to be useful again [7].

Considering the length of the Netflix sessions, we have found that the majority of users use this application to see videos with a length between 40 and 45 minutes. The duration of an episode of a series is this length, hence this results show that users mainly use Netflix to follow their favorite series. The second most probable length was around 90 minutes; this length can be related to the length of a movie, which is normally 90-120 minutes.

We can see that the importance of VOD systems in today's Internet traffic, for that reason an improvement in their architecture and data delivery will improve considerably the QoS of a majority of Internet users.

On the other hand, the popularity of a video game was found to be related to its renovation; if users consider the changes done interesting or not. Hence, users' experience is harder to improve just considering an enhancement in Internet traffic, the main responsible of the improvement is the creator of video games.

References

[1] J. Li, A. Aurelius, V. Nordell, M. Du, Å. Arvidsson and M. Kihl, "A five year perspective of traffic pattern evolution in a residential broadband access network," *Future Network & Mobile Summit 2012*

[2] A. Aurelius, C. Lagerstedt and M. Kihl, "Streaming media over the Internet: Flow based analysis in live access networks," *IEEE International Symposium on Broadband Multimedia Systems and Broadcasting*, Nuremberg, Germany, 8-10 June 2011

[3] J. Andersson, T.Bonnedahl, S. Höst and M. Kihl, "Traffic patterns on different Internet access technologies," *Proceedings of SNCNW+Adhoc 2009*, The 6th Swedish National Computer Networking Workshop and The 9th Scnadinavian Workshop on Wireless Adhoc Networks, Uppsala, pp. 129-133, 2009-05-04/2009-05-05.

[4] A. Ali-Eldin, J.Tordsson, M.Kihl and E.Elmroth, "Analysis and characterization of a video-on-demand service workload," *MMSys'15*, March 18 - 20 2015, Portland, OR, USA.

[5] H.Yu, D.Zheng, B.Zhao and W.Zheng, "Understanding user behaviour in a large-scale video-on-demand systems," *EuroSys'06*, April 18–21, 2006, Leuven, Belgium.

[6] M. Du, M. Kihl, Å. Arvidsson, H. Zhang, C. Lagerstedt and A. Gavler, "Prefetching Schemes and Performance Analysis for TV on Demand Services," *International Journal on Advances in Telecommunications*, Vol. 8, No. 3&4, pp. 162-172, 2015.

[7] M. Forzati and C. Larsen, "On the symmetry requirements for tomorrow's fibre access networks," in *11th International Conference on Transparent Optical Networks*, 2009.

[8] Procera Networks, http://www.proceranetworks.com

[9] Datastream Recognition Definition Language (DRDL), http://documents.sysob.com/procera_Intro_DRDL.pdf

[10] O. Cappe, , Xueshi Yang (May 2002). Long range dependence and heavy-tail modeling for teletraffic data. IEEE signal processing magazine. Vol: 19 issue: 3 , pages: 14 –27

[11] Leland, W. E., Taqqu, M. S., Willinger, W., and Wilson, D. V. 1994. "On the self-similar nature of Ethernet traffic (extended version)," IEEE/ACM Trans. Netw. 2, 1 (Feb. 1994), 1-15