Ultra TV

Deliverable 3.3
QoE Evaluation in adaptive streaming scenarios

Version: V0.3
Date: 27/03/2017
Distribution: Consortium Confidential
Responsible: IT-Instituto de Telecomunicações
Contributor: IT-Instituto de Telecomunicações
## Document History

<table>
<thead>
<tr>
<th>Version</th>
<th>Date</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>V0.1</td>
<td>02/03/2017</td>
<td>Contributions</td>
</tr>
<tr>
<td>V0.2</td>
<td>20/03/2017</td>
<td>Revisions</td>
</tr>
<tr>
<td>V0.3</td>
<td>27/03/2017</td>
<td>Final Version</td>
</tr>
</tbody>
</table>
1 Executive Summary

In this document an overview is taking into quality of experience (QoE) solutions for modern streaming technologies and protocols, which use CDNs as content distribution infrastructures, with a particular emphasis on adaptive and scalable streaming algorithms. Expanding on these topics, a focused examination is conducted on Over-The-Top (OTT) multimedia caching technologies and solutions, tailored towards prefetching algorithms.

To understand the research challenges of QoE adaptive streaming in OTT delivery networks a deep insight is required on their usage scenarios and supporting technologies. By considering the need to maintain high levels of QoE, and the challenges imposed by adaptive streaming technologies, it is necessary to acquire knowledge in QoE techniques (subjective and objective) and QoE solutions in over-the-top video networks. Therefore, a closer look is given on QoE Impact and generalized requirements for adaptive streaming, as well as in QoE estimation and optimization on HTTP adaptive streaming.

Content Caching management and distributed cache are also overviewed comprising algorithms used to content replacement. The algorithms approaches are explained and some distributed caching content solutions using these algorithms are presented.

At the end, we include an overview focusing on web prefetching algorithms, such as prefetch by Popularity, Lifetime and Good Fetch. The algorithms presented only consider as decision metrics the characteristics of the web objects, emphasizing the need for a new approach that takes into account the consumer’s behavior and the content itself.
### Acronyms

<table>
<thead>
<tr>
<th>Acronym</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AAC</td>
<td>Advanced Audio Coding</td>
</tr>
<tr>
<td>ABS</td>
<td>Adaptive Bitrate Streaming</td>
</tr>
<tr>
<td>AES</td>
<td>Advanced Encryption Standard</td>
</tr>
<tr>
<td>AIR</td>
<td>Adobe Integrated Runtime</td>
</tr>
<tr>
<td>ATS</td>
<td>Apache Traffic Server</td>
</tr>
<tr>
<td>AVC</td>
<td>Advanced Video Coding</td>
</tr>
<tr>
<td>AVS</td>
<td>Accenture Video Solution</td>
</tr>
<tr>
<td>CAE</td>
<td>Content Adaptation Engine</td>
</tr>
<tr>
<td>CARP</td>
<td>Cache Array Routing Protocol</td>
</tr>
<tr>
<td>CDN</td>
<td>Content Delivery Network</td>
</tr>
<tr>
<td>CDNI</td>
<td>Content Delivery Network Interconnection</td>
</tr>
<tr>
<td>CDS</td>
<td>Content Delivery System</td>
</tr>
<tr>
<td>CPS</td>
<td>Cicero Provisioning Server</td>
</tr>
<tr>
<td>DECE</td>
<td>Digital Entertainment Content Ecosystem</td>
</tr>
<tr>
<td>DHT</td>
<td>Distributed Hash Tables</td>
</tr>
<tr>
<td>DM</td>
<td>Device Management</td>
</tr>
<tr>
<td>DNS</td>
<td>Domain Name Server</td>
</tr>
<tr>
<td>DPI</td>
<td>Deep Packet Inspection</td>
</tr>
<tr>
<td>DRM</td>
<td>Digital Rights Management</td>
</tr>
<tr>
<td>DVR</td>
<td>Digital Video Recorder</td>
</tr>
<tr>
<td>EME</td>
<td>Encrypted Media Extensions</td>
</tr>
<tr>
<td>FIFO</td>
<td>First in, first out</td>
</tr>
<tr>
<td>FTTH</td>
<td>Fiber-To-The-Home</td>
</tr>
<tr>
<td>GOP</td>
<td>Group of Pictures</td>
</tr>
<tr>
<td>GPL</td>
<td>General Public License</td>
</tr>
<tr>
<td>HAS</td>
<td>HTTP Adaptive Streaming</td>
</tr>
<tr>
<td>HDS</td>
<td>HTTP Dynamic Streaming</td>
</tr>
<tr>
<td>HLS</td>
<td>HTTP Live Streaming</td>
</tr>
<tr>
<td>HTCP</td>
<td>HyperText Caching Protocol</td>
</tr>
<tr>
<td>HW</td>
<td>Hammerstein-Wiener</td>
</tr>
<tr>
<td>IETF</td>
<td>Internet Engineering Task Force</td>
</tr>
<tr>
<td>IIR</td>
<td>Infinite Impulse Response</td>
</tr>
<tr>
<td>IIS</td>
<td>Internet Information Services</td>
</tr>
<tr>
<td>ILP</td>
<td>Integer Linear Programming</td>
</tr>
<tr>
<td>IP</td>
<td>Internet Protocol</td>
</tr>
<tr>
<td>KQI</td>
<td>Key Quality Indicators</td>
</tr>
<tr>
<td>LFU</td>
<td>Least Frequently Used</td>
</tr>
<tr>
<td>LRU</td>
<td>Least Recently Used</td>
</tr>
<tr>
<td>MCNKP</td>
<td>Multiple-Choice Nested Knapsack Problem</td>
</tr>
<tr>
<td>MDC</td>
<td>Multiple Description Coding</td>
</tr>
<tr>
<td>MOS</td>
<td>Mean Opinion Score</td>
</tr>
<tr>
<td>MPD</td>
<td>Media Presentation Description</td>
</tr>
<tr>
<td>MPEG</td>
<td>Moving Pictures Expert Group</td>
</tr>
<tr>
<td>MPU</td>
<td>Most Popularly Used</td>
</tr>
<tr>
<td>MSE</td>
<td>Media Source Extensions</td>
</tr>
<tr>
<td>NAT</td>
<td>Network Address Translation</td>
</tr>
<tr>
<td>NMS</td>
<td>Network Management System</td>
</tr>
</tbody>
</table>
Deliverable 3.3

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Full Form</th>
</tr>
</thead>
<tbody>
<tr>
<td>NP</td>
<td>Non-deterministic Polynomial-time</td>
</tr>
<tr>
<td>OMA</td>
<td>Open Mobile Alliance</td>
</tr>
<tr>
<td>OPEX</td>
<td>Operational Expenditures</td>
</tr>
<tr>
<td>OS</td>
<td>Operating System</td>
</tr>
<tr>
<td>OTT</td>
<td>Over-the-Top</td>
</tr>
<tr>
<td>PIFF</td>
<td>Protected Interoperable File Format</td>
</tr>
<tr>
<td>PSNR</td>
<td>Peak Signal-to-Noise Ratio</td>
</tr>
<tr>
<td>PSS</td>
<td>Packet-switched Streaming Service</td>
</tr>
<tr>
<td>QMON</td>
<td>Quality Monitoring</td>
</tr>
<tr>
<td>QoE</td>
<td>Quality of experience</td>
</tr>
<tr>
<td>QoS</td>
<td>Quality-of-Service</td>
</tr>
<tr>
<td>RAN</td>
<td>Radio Access Network</td>
</tr>
<tr>
<td>RNN</td>
<td>Random Neural Networks</td>
</tr>
<tr>
<td>RSTP</td>
<td>Real Streaming Transport Protocol</td>
</tr>
<tr>
<td>SLA</td>
<td>Service Level Agreements</td>
</tr>
<tr>
<td>SSIM</td>
<td>Structural Similarity</td>
</tr>
<tr>
<td>STB</td>
<td>Set-Top-Boxes</td>
</tr>
<tr>
<td>STSQ</td>
<td>Short-Time Subjective Quality</td>
</tr>
<tr>
<td>SVC</td>
<td>Scalable Video Coding</td>
</tr>
<tr>
<td>TVSQ</td>
<td>Time-Varying Subjective Quality</td>
</tr>
<tr>
<td>UI</td>
<td>User Interface</td>
</tr>
<tr>
<td>VCAS</td>
<td>Video Content Authority System</td>
</tr>
<tr>
<td>VCR</td>
<td>Video Cassette Recorder</td>
</tr>
<tr>
<td>VoD</td>
<td>Video-on-Demand</td>
</tr>
</tbody>
</table>
# Table of Contents

1 Executive Summary .................................................................................................................. 4

2 Quality-of-Experience (QoE) ..................................................................................................... 12
   2.1 QoE Evaluation Techniques ................................................................................................. 12
       2.1.1 Subjective Assessment ............................................................................................... 12
       2.1.2 Objective Assessment ............................................................................................... 13
       2.1.3 Hybrid Assessment ..................................................................................................... 13
   2.2 QoE on Over-the-Top (OTT) video networks ................................................................. 14
   2.3 QoE in Adaptive Streaming solutions ................................................................................ 15
       2.3.1 QoE in 3GPP DASH ................................................................................................. 16
   2.4 QoE Impact and generalized requirements for Adaptive Streaming ............................. 18
   2.5 QoE Estimation and Optimization on HTTP Adaptive Streaming ............................... 19
       2.5.1 QoE Estimation on HAS ............................................................................................ 19
       2.5.2 QoE Optimization on HAS ...................................................................................... 21
       2.5.3 Optimization through content preparation ............................................................... 22
       2.5.4 Optimizing the delivery process ................................................................................. 23
       2.5.5 Optimizing client adaptation mechanisms ................................................................. 23

3 Content Distribution .................................................................................................................. 25
   3.1 Typical Content Delivery Network Architecture .......................................................... 25
   3.2 Content Distribution Architectures .................................................................................. 26
       3.2.1 Centralized Content Delivery .................................................................................... 26
       3.2.2 Proxy-Caching .......................................................................................................... 27
       3.2.3 Peer-to-Peer (P2P) .................................................................................................... 28
       3.2.4 Hybrid Delivery ........................................................................................................ 28
       3.2.5 N-Tier Caching .......................................................................................................... 29
       3.2.6 Content Delivery and Management System ............................................................. 30
       3.2.6.1 Replica Server Placement ...................................................................................... 30
   3.3 Content Delivery Network Interconnection (CDNI) ...................................................... 31
   3.4 Mobile Content Delivery Networks (mCDNs) .............................................................. 33
   3.5 CDNs and Multimedia Streaming ..................................................................................... 34
   3.6 OTT Multimedia Delivery ................................................................................................. 34
       3.6.1 Content Delivery Pipeline .......................................................................................... 35
       3.6.1.1 Content Preparation ............................................................................................. 36
       3.6.1.2 Content Ingest ....................................................................................................... 36
       3.6.1.3 Content Distribution ............................................................................................. 36
       3.6.1.4 Consumption .......................................................................................................... 37
   3.7 OTT Multimedia Services of Telecommunication Operators ....................................... 37
       3.7.1 Linear TV .................................................................................................................... 37
       3.7.2 Time-shift TV ............................................................................................................. 37
       3.7.3 Video-on-Demand (VoD) .......................................................................................... 38
   3.8 Impact of Catch-up TV Services ....................................................................................... 38

QoE Evaluation in adaptive streaming scenarios 7
Deliverable 3.3

3.9 Usage of Catch-up TV Services

3.10 HTTP Adaptive Streaming

3.10.1 Source Video Coding

3.10.1.1 Multiple Description Coding (MDC)

3.10.2 Scalable Video Coding (SVC)

3.11 Adaptive Segmented HTTP-based delivery

3.11.1 Why HTTP Delivery?

3.11.1.1 Microsoft Smooth Streaming

3.11.1.2 Apple HTTP Live Streaming (HLS)

3.11.1.3 Adobe HTTP Dynamic Streaming (HDS)

3.11.1.4 MPEG Dynamic Adaptive Streaming over HTTP (DASH)

3.12 Content Cache OTT Delivery: Multicast and Unicast Solutions

4 Cache Management and Distributed Cache Solutions

4.1 Caching Algorithms

4.2 Distributed cache content

4.2.1 Redis

4.2.2 Memcached

4.2.3 Cachetol

5 Content Prefetching

5.1 Prefetch Techniques

5.1.1 Prefetch by Popularity

5.1.2 Prefetch by Lifetime

5.1.3 Good Fetch/Threshold

5.2 Replica Servers and Prefetching

5.3 Web Servers and Content Caching Technologies

5.4 Embedded Proxy-Cache Solutions

6 Conclusions
# List of Figures

1. QoE vs QoS Scope .......................................................... 14
2. Framework for modeling the QoE of networked services .................. 15
3. Hammerstein-Wiener model for TVSQ prediction [22] ...................... 21
4. QoE optimization aspects on HTTP Adaptive Streaming .................. 22
5. Typical Content Delivery Network Architecture .......................... 25
6. Centralized OTT Delivery .................................................. 26
7. Proxy Cache OTT Delivery ................................................. 27
8. P2P OTT Delivery .......................................................... 28
9. Hybrid OTT Delivery ....................................................... 29
10. Example of 2 Tier Caching Architecture ................................... 30
11. CDNI use case 1 ........................................................... 32
12. CDNI use case 2 ........................................................... 32
13. Typical mobile CDN infrastructure ......................................... 34
14. Content Delivery Pipeline Example ......................................... 36
15. Adaptive Streaming Classes ................................................ 41
16. Multiple Description Coding Example. Adapted from [93] ................. 42
17. Scalability modes of Scalable Video Coding (SVC) ......................... 42
19. Simplified Smooth Streaming Session Diagram ............................ 46
20. Sample Smooth Streaming Server Manifest .................................. 46
21. Sample Smooth Streaming Client Manifest .................................. 47
22. Sample Apple HLS M3U8 Top Level Playlist ............................... 47
23. Sample Apple HLS M3U8 Track Playlist ................................... 47
25. MPEG-DASH Media Presentation Description (MPD) Hierarchy [98] .... 49
26. Accenture's Video Solution architecture ...................................... 51
27. Alcatel Lucent's Velocix solution architecture ................................ 51
28. Broadpeak's NanoCDN solution architecture ................................ 52
29. Cicero's solution architecture .............................................. 53
30. Ciscos' Videoscape solution architecture .................................... 53
31. Edgeware's solution architecture ............................................ 54
32. Envivio/Verimatrix's solution architecture .................................... 55
33. Octoshape's solution architecture .......................................... 56
34. Pace's solution architecture ................................................. 56
35. Smartlab's solution architecture ............................................. 57
36. Key-value Data Model ...................................................... 60
37. Redis Architecture example ................................................. 61
38. W/WO Memcache Architecture ............................................ 62
List of Tables

Table 1 - FIFO cache Replacement Policy ................................................................. 58
Table 2 - LRU cache Replacement Policy ................................................................. 58
Table 3 - MPU cache Replacement Policy ................................................................. 59
Table 4 - Comparison of Proxy Cache Solutions ......................................................... 66
Table 5 - Comparison of Five Proxy Cache Solutions - Main features ......................... 67
List of Equations

Equation 1 – eMOS estimation ................................................................. 21
2 Quality-of-Experience (QoE)

2.1 QoE Evaluation Techniques

The typical network performance analysis is covered by QoS. However, nowadays given the OTT content and the highly user-centered content, QoE is taking an important part in the network analysis. It is crucial to understand which factors can be controlled and which cannot in order to maximize the users’ QoE. To do so, there must be a way to measure QoE, since something that cannot be measured, cannot be optimized. QoE measurement, or assessment, is divided into two main categories, depending on how it is performed: subjective and objective.

2.1.1 Subjective Assessment

Subjective QoE assessments involve surveys to people that have been exposed to the service or application whose QoE is being assessed. These surveys rely on users’ opinions to rate the service performing under different conditions. The rating system may be based on qualitative or quantitative evaluations. Also this evaluation technique typically requires users to give their opinion which will be highly based on personal life experiences giving a different perspective by each different user, its mood, or perhaps the system responsiveness.

Qualitative evaluations tend to focus on comparative evaluations between different experiments, such as indicating that the first experience was more pleasant than the second one. This type of evaluation may require a huge post processing of the data, perhaps with advanced statistical tools, to be able to be used in computer simulations or operations, as it is not formatted for a computer to understand. As such, it does not appear to be the best option to be used in a dynamic QoE environment that needs to adapt itself to multiple constraints. An example would be a user answering an open-ended question comparing the assessed service or application.

Some of the examples would be:

- What do you think was worse/better in the second compared video?
- Which video you consider better, the first or the second?
- How the user experience improved/decreased?
- What is your opinion on the application/system?
- What do you think could be improved?

As for quantitative assessments, users are asked to use a number to grade their experience according to pre-established scales; thus, being objective. This is more widely used as it facilitates data processing. In the context of video reproduction, ITU-R recommends the usage of BT.500–13 [1] for video quality assessment. Given the pre-defined scales, a computer can use them as values which can then be used to adjust the so called dynamic adaptive QoE environment as needed. As such, this appears to be most suited for the intended goal under the subjective type.

Some examples would be:

- On a scale of 1 to 5, how would you rate the first/second video?
- Using the same scale, how would you rate the system?
- Give a score to the experience.
- Score the video quality.
2.1.2 Objective Assessment

In contrast to the previously described subjective assessment methods, which are laborious, time-consuming and expensive, the objective approach is an alternative that builds on the technical characteristics usually provided by QoS parameters to model and extrapolate a perceptual quality indicator. Because there is usually no way of ensuring that, by itself, the model is accurate, objective assessment algorithms are usually trained and modeled by resorting to known QoE estimations from subjective models.

There are some objective metrics that may be used, such as video frame rate, resolution, and compression level, which can be monitored using specific tools [2], and then correlated with the users' perceived quality. These correlations enable the creation of QoE models based on technical parameters.

Some examples would be:

- Amount of buffering (milliseconds);
- Average throughput (Mbps);
- Number of chunks requested on each quality level;
- Number of times the quality increased/decreased;
- Total consumed time vs requested time;
- Initial playout delay;
- Seeking delay;

According to [3], there has been some development in the Content Inspection/Artifact-based Measurement. These two are related to the artifact analysis of the video and operate over the decoded content which is a way to measure QoE. This is a quite exact way to measure one of the parameters for QoE. This method appears to be quite CPU intensive, because requires further analysis to evaluate its possible advantages to the intended purpose.

Some other way to have objective metric, which some providers actually use, is to install a device that is able to measure the metrics at the users' side. This can be done by means of some middlebox to allow direct investigation and influence of traffic conditions. However this method will rapidly prove itself highly challenging due to the addition of end-to-end encryption [4].

Naturally, how objective QoE assessments are performed depends strongly on the service under consideration. In the case of uncontrolled OTT networks, there are QoS and QoE-specific metrics that may be considered when focusing on OTT video streaming.

These metrics can be studied and, using computer simulations or algorithms, the network can be dynamically changed to have a rule on the QoE. For example in [5], using only Buffer and Bit Rate information, it was possible to study in detail on how to improve a DASH stream performance and its impact on QoE.

2.1.3 Hybrid Assessment

Quantifying the quality of video being transmitted over heterogeneous networks is becoming critical in the field of digital video Communication. Thus, hybrid solutions turn be an alternative, which involves both subjective and objective techniques. A possible way to improve QoE could be the usage of a neural network model that would be responsible for estimating the relation between QoS parameters and Qo, thus making sure that it would provide a positive final result to QoE [6]. This has been called a hybrid technique since it can be based on both aforementioned methods. It has a high
potential of improving overall QoE, but given the blackbox characteristic of Neural Networks, it shows very difficult to study and replicate the same results under different conditions.

### 2.2 QoE on Over-the-Top (OTT) video networks

There has been a growing scientific interest in QoE over the last decade motivated by the multidimensional characteristics of the human experience in technology interaction [7]. However, common scientific approaches tend to ultimately focus on Quality-of-Service (QoS) metrics under the assumption that mere improvements in QoS will lead to improved QoE, disregarding user-related aspects such as expectations, or previous experiences.

This perspective has been supported by the latest technological advancements in content delivery technologies, tools for developers, and access networks (such as FTTH, LTE, etc), which usually improve the overall QoS and, to some extent, the QoE. Nevertheless, it is a well-know fact [8] that QoE is the key factor that should be looked into, as users have high-quality expectations that, if not met, might jeopardize their loyalty.

QoE is defined by International Telegraph Union Telecommunication Standardization Sector (ITU-T) [9] as:

> The overall acceptability of an application or service, as perceived subjectively by the end-user.

There is a departure from the traditional QoS perspective which only encompasses the networking aspects of the services such as [8,10]: throughput; goodput; delay; and loss ratio. The description of QoS, according to ITU-T, is:

> The totality of characteristics of a telecommunications service that bear on its ability to satisfy stated and implied needs of the user of the service.

Kilkki [8] clearly differentiated the two concepts with the following statement:

> It is quite meaningless to say that the goal of network operations is high QoS (a similar statement would be to claim that the purpose of life is to speak perfect English).

A user performing web-browsing is not concerned with the loss ratio of the connection. He only cares about opening the web-page he was looking for in a reasonable amount of time. Figure 1 illustrates the different scopes of QoE vs QoS. QoE reflects a different perspective on quality monitoring and tries to answer the why question: why is the video stuttering? why does the user feel frustrated? [8,11].

![Figure 1 - QoE vs QoS Scope.](image)

Figure 2 is taken from [12], where the logarithmic laws in quality perception and the complex relationship between factors that ultimately culminate in the user’s QoE are evaluated. There are external factors such as users’ personalities, the application/service usability, and the usage context, to name a few; and then there are service-related Key Quality Indicators (KQIs) that also influence the users’ QoE.
QoE Evaluation in adaptive streaming scenarios

2.3 QoE in Adaptive Streaming solutions

Adaptive streaming technologies aim to improve the QoE of the streamed video over time by relinquishing some degrees of control to the end client, which may then adapt to changing conditions and minimize rebuffering events.

The client is in a unique position to assess its environment conditions and must be able to decide which stream to consume, from a set of server-provided alternative streams, each with different video and/or audio characteristics. To that end, several parameters must be modeled, estimated, and monitored, such as:

- **Network Resources** - Bandwidth, Delay, Jitter, Availability;
- **Capabilities** - Available Memory, CPU, Screen Resolution, Remaining Power;
- **Streaming conditions** - Buffer Size, Desired seek speed, Desired start-up delay.

These extra degrees of control add to the number of dimensions contributing to a good user QoE, and despite having the potential to ultimately benefit QoE, they may very well hinder it if the client control algorithms are not adequately tuned.

Scenarios where there is “enough bandwidth, but not enough CPU power for decoding high-bitrates”, or there is “good bandwidth, available CPU and memory, but low remaining battery power”, are commonplace, and must be accounted for.

An extensive QoE study is performed in [13] where it is shown that the crucial QoE advantage provided by HTTP Adaptive Streaming (HAS) when compared to progressive streaming is mostly due to the reduction of rebuffering events or stalls. This metric has been shown to have a critical impact on
Deliverable 3.3

the Mean Opinion Score (MOS). In addition to these findings, relationships are also established between QoE and factors like quality switching frequency, initial playout delay, startup bit rate and average bit rate. All of these factors must be weighted in order to maximize the users' QoE.

Due to the novelty of this technology, when compared to the more traditional push-based adaptive streaming techniques, several challenges and opportunities arise. One of the crucial-for-success challenge is the development of adequate methodologies and metrics for assessing the users' QoE for adaptive streaming services.

Realizing this need, both 3rd Generation Partnership Project (3GPP) [14] and Moving Pictures Expert Group (MPEG) [14,15] bodies identified QoE metrics for Dynamic Adaptive Streaming over HTTP (DASH), which apply to adaptive streaming technologies in general. The 3GPP proposal also specifies methods for QoE reporting back to the network servers which may provide crucial insights.

Monitoring the QoE is highly beneficial for debugging failures, managing streaming performance, improving client adaptation technologies, and also to provide valuable input to resource provisioning systems.

2.3.1 QoE in 3GPP DASH

3GPP and MPEG identified QoE performance metrics and reporting protocols as playing a critical role in optimizing the delivery of Adaptive Streaming services, and have thus considered them in their DASH specification.

3GPP’s TS 26.247 [14] specification is quite detailed and includes mechanisms for triggering client-side QoE measurements along with the specification of protocols for reporting them back to the server. 3GPP mandates that client devices supporting QoE features (an optional requirement) have to support the full set of the requested metrics.

The QoE reporting feature is mainly comprised of three stages. In the first one, the trigger phase, the server requests QoE reports from the clients by using either the Media Presentation Description (MPD) or the Open Mobile Alliance (OMA) Device Management (DM) QoE Management Object to specify a percentage of clients that should activate the QoE reporting features. The clients will then use a local random number generator to decide whether they fall into the specified percentage of devices that should report the metrics.

The next phase regards the actual gathering of QoE information, which happens according to the configuration specified in the MPD or OMA DM.

Finally, the client reports the metrics back to a network server.

An extensive amount of metrics are collected so that the servers monitoring the QoE of their clients are able to accurately estimate the users' QoE. Although the focus of this section is on Adaptive Streaming, 3GPP’s TS 26.247 also specifies a subset of QoE metrics that should be used in the event of progressive streaming sessions.

Regarding the specified Adaptive Streaming QoE metrics, they are as follows:

1. HTTP Request/Response Transactions

The client must provide a list of all HTTP requests and responses finished within the QoE metric collection period, specifying the following metrics:

a. Type of request, (MPD, MediaSegment, ...);
b. Request url and actual url if any redirect was performed;
c. HTTP response code;
d. byte-range-spec part of the HTTP Range header;
e. Request timing information (time at which the request was sent and the response received);
   f. Throughput trace information for successful requests.

2. **Representation Switch Events**

A switch event is triggered when the first HTTP request for a new representation is sent. It represents a client decision on the representation that should be reproduced.

a. Time of switch event;
   b. Media time of the earliest media sample played out from the new representation;
   c. Representation Id;
   d. SubRepresentation Level.

3. **Average Throughput**

Report of the average throughput observed by the client during the measurement interval.

a. Total number of bytes in the body of HTTP responses received;
   b. Activity time in milliseconds (i.e. excluding inactivity periods);
   c. Start time of the measurement interval;
   d. Measurement duration;
   e. Access Bearer for the TCP connection for which the average throughput is reported;
   f. Inactivity type (pause, buffering, . . . ) if known and consistent in the report period.

4. **Initial Playout Delay**

The initial playout delay is considered to be the time elapsed between fetching the first media segment and retrieving that segment from the client buffer for playback.

5. **Buffer Level**

Reports a list of buffer level status events measured during playout at normal speed.

a. Time of the measurement;
   b. Buffer level in milliseconds.

6. **Play List**

Contains a list of playback periods. A playback period is defined as the time interval between a given user action and whichever occurs soonest: a subsequent user action; the end of playback; or a failure that stops playback.

a. Timestamp of the user action that triggered the playback period;
   b. The media (presentation) time at which the playout was requested;
   c. The action type that triggered the playback period (initial playback, seek, resume, user requested quality change, . . .);
   d. Trace of played segments, containing their RepresentationId, SubRepresentation level, timing, playback speed, stop reason, and duration. The trace may contain entries for different representations that overlap in time, due to different representations being played simultaneously (e.g. audio and video).

7. **MPD Information**

In order to provide adequate information to servers that may not have access to the MPD, this information must be sent whenever any other metric references a Representation for which MPD information has not been reported yet, so that the servers have sufficient information on the media characteristics.

a. Representation Id addressed by the QoE metrics report;
   b. SubRepresentation level addressed. If not present, the report concerns the complete representation;
2.4 QoE Impact and generalized requirements for Adaptive Streaming

There are two commonly used types of QoE representation models, depending on the dimensions considered: pixel-domain models, and bit stream-domain models.

The pixel domain models may be further subdivided into 3 main categories, according to the information required by the models' algorithms:

1. Full-Reference Models - require the complete original reference video for comparison; they provide high accuracy and repeatability at the expense of intense processing and/or bandwidth;

2. Reduced-Reference Models - require a partial view on the reference video. They use features extracted from the original video to perform the comparison. For instance, they trade-off bandwidth for the reference signal with measurement accuracy;

3. No-Reference Models - rely only on the degraded signal to perform a quality estimation; hence, the estimates are less accurate. The reference signal is unknown. The second type of models - bit stream models - inspect the video flow and use the extracted parameters to infer QoE. Relevant parameters include: flow bit rate, packet losses, jitter, and RTT. In these models, the video is not effectively decoded. As far as video image quality is concerned, popular metrics using full reference models include Peak Signal-to-Noise Ratio (PSNR), and Mean Squared Error (MSE); however, they serve merely as indicators given that under some particular circumstances the results may be deceiving [203]. Nevertheless, on most situations they do provide valuable insight on the image quality.

When transmitting video over TCP/IP, using progressive streaming for example, other factors must be taken into account that influence QoE.

Due to the reliable nature of TCP, the issue of lost frames does not present itself as it does in on UDP based streaming; thus, the variations of QoE in this scenario are usually related to network delays and buffering issues. If the buffer is being filled at a slower pace than it's being consumed, the playback will frequently have to stop and wait for more data. Because these stops have a large impact on the perceived QoE, the authors of [160] have proposed a focus on the temporal structure of the videos being streamed, which culminated into 3 main metrics:

- Initial buffering delay - Time delay until initial playback is started;
- Average rebuffering duration - How long rebuffering events last;
- Rebuffering frequency - How often rebuffering events occur;

Similar conclusions were drawn in [16], where an aggregate metric called pause intensity combines both the average rebuffering duration and its frequency. The results attained showed a remarkable correlation between pause intensity and MOS variability.

It is also shown in [17] that the initial buffering delay does not have a large impact in the QoE; the users would rather wait a bit more for the playback to start, than have rebuffering events throughout the playback session. Although this is a valid assumption on VoD services such as Netflix, in the context of live broadcast streaming, the users are more sensitive to initial-buffering delays, as they expect the stream to start right away.
Both studies also concluded that in the particular context of streaming over TCP/IP, the temporal aspects of the video playback have higher impact on QoE than spatial artifacts, such as macroblocks, or blurriness.

2.5 QoE Estimation and Optimization on HTTP Adaptive Streaming

2.5.1 QoE Estimation on HAS

Understanding how QoE may be estimated and how it can be improved along the content distribution pipeline is of paramount importance; therefore, a literature analysis is required on methods for estimating QoE.

Performing a subjective study of QoE for a given video playback session is costly and cannot be performed in real time; thus, automatic methods for estimating QoE are valuable and desirable. The authors of [18] realized this issue and proposed an adaptation of Pseudo-Subjective Quality Assessment (PSQA) [19] to HAS using H.264/Advanced Video Coding (AVC). The created no-reference QoE estimation module based on Random Neural Networks (RNNs) was able to provide fair estimates on the QoE of 18 validation videos. However, some restrictions were imposed, namely on the analyzed quality dimensions, as the estimation module was limited to the Quantization Parameter, used as an indicator of video compression and bit rate, and on a model for playout interruptions.

Despite being able to capture important metrics and conveying them into a QoE estimate, this QoE estimation module fails to encompass other metrics identified as playing a significant role on HAS QoE such as quality switching, playback resolution, or even the initial playout delay [13].

In another study [20] the issue of QoE estimation was addressed in the context of Radio Access Networks (RAN). Two different base video clips were prepared to target tablets (iPads) and smartphones (iPhones). Using these base video clips as HLS sources, several LTE network conditions were simulated to create different playback scenarios. The resulting video clips were reconstructed at the client devices in order to allow for offline and comparable MOS evaluations for each of the 90 reconstructed clips, which were later evaluated by a total of 500 volunteers.

The subjective evaluations were then used as the ground truth for MOS linear prediction models where the quality metrics considered were Peak Signal-to-Noise Ratio (PSNR), Structural Similarity (SSIM), nominal chunk bit rate, and chunk-MOS - where a given MOS is associated with a specific chunk quality level.

The model's result show that the bit rate based model is the worst, which can be explained by the non-linear relationship between bit rate and MOS. The PSNR and SSIM approaches provided fair results, while chunk-MOS’s performance was shown to be the best, especially with regard to sensitivity to non-optimal model parameters. These characteristics make the chunk-MOS approach suitable for classifying unseen content.

The authors state that the impact of quality level switches is not significant in their test, even though other studies [13] clearly demonstrate its impact on QoE.

A different approach is taken in [21], where the authors of Quality Monitoring (QMON) take on the QoE estimation issue with a network-monitoring approach relying on Deep Packet Inspection (DPI), by placing an intermediate buffer proxy between the video source and the client. Given the network proxy approach, no direct access to the client devices is required. Despite being focused on
progressive streaming, and not addressing HAS directly, a method for extending QMON to support HAS is suggested.

The MOS estimation relies solely on a buffer stalling evaluation. The buffer stalls are weighted in a “negative impact” exponential function that aims to capture the aggravated impact on QoE at each subsequent stall event, along with the duration of each stall. The network monitor relies on buffer fill level estimations based on timestamps embedded on the video payload of a given TCP flow and has 3 modes of operation.

The first one, the exact method, decodes every TCP packet to try and extract the video timestamp and compare it with the timestamp of the respective TCP segment. Because every single packet is decoded, this is the most accurate method, at the expense of intense computational requirements.

As for the second approach, the estimation method, increases the processing speed of QMON by fully decoding only the video header and extracting the size and duration of all subparts, which are then used as baselines for estimating the buffer fill level relying solely on the amount of TCP data streamed. This is a fair approach if the network does not experience a significant number of TCP retransmissions, as these add up to the amount of data streamed and influence the buffer fill level estimation.

Lastly, the combined method uses the previous methods dynamically to adapt to the experienced transport conditions. If a significant amount of TCP retransmissions is experienced, the exact method is used, otherwise the estimation method is preferred.

The performance evaluation of the buffer fill level estimation is shown to be accurate on both the exact and combined method, with the estimation method falling behind.

Regarding the MOS estimation, which is the focus of the paper, despite having provided a proposal for an estimation metric, the provided metric is not validated with any test subjects; thus, its accuracy is questionable.

In [22], a Time-Varying Subjective Quality (TVSQ) computation is performed relying on the videos’ estimated Short-Time Subjective Quality (STSQ). As the authors put it, TVSQ “is a time series or temporal record of viewers’ judgments of the quality of the video as it is being played and viewed”, while STSQ is a “scalar prediction of viewers’ subjective judgment of a short video’s overall perceptual quality”. The dynamic model is defined to consider the quality-variation aspects of HAS in such a manner that online QoE evaluation of test videos is feasible.

This approach stems from the fact that video watching is a time-varying experience, with a non-linear relationship between the current frame quality and the current experience quality, due to the viewers’ memory effect.

In order to create the STSQ predictions, the Video-RRED [23] algorithm is used due to its good prediction accuracy and performance. Then, the STSQ inputs are fed into a dynamic system model of TVSQ, which then outputs a prediction of TVSQ.

The TVSQ system proposed is an Hammerstein-Wiener (HW) model with generalized sigmoid input and output functions capable of capturing the non-linearities between the input STSQ and the output TVSQ, and an intermediate linear Infinite Impulse Response (IIR) filter, as shown in Figure 3. The model’s parameters are then estimated with the help of reference TVSQ generated from subjective video-viewing tests.

In the validation phase, the “leave-one-out” cross-validation approach is taken, i.e. all the reference videos except one are used to train the model, and the model is then applied to the validation video. The results show a high linear and Spearman’s rank correlation coefficient (~0.9) between the measured TVSQ and the estimated one, and appear to be robust for the tested data set. With regard to the stability of online predictions of TVSQ, the model is shown to be stable in the presence of untrained videos.
The proposed TVSQ-based approach greatly succeeds in modeling non-linearities and the chunk-base nature of HAS, providing a valuable contribution to the estimation of QoE in HAS sessions. Some aspects such as initial playout delay are not addressed, and the practical feasibility may be not be ideal given the STSQ estimation approach taken, however, with adjustments to the way how the STSQ are computed, this model demonstrates an excellent potential for accurate, online, QoE estimation.

An alternative for the previous one is considered in [24]. The authors focus on the DASH variant of HAS, and identify a set of key QoE-impacting metrics. The buffer overflow/underflow events are analyzed with respect to their re-buffering frequency and average duration; the frequency and amplitude of quality switches are also considered, given their proven impact on QoE [25]; and, lastly, an objective content quality level metric is examined which encompasses factors such as video bit rate, frame rate and quantization parameter.

Taking the previous parameters into consideration, An eMOS (estimated MOS) analytical exponential model is developed and is later calibrated through subjective testing. The model is presented in Equation 1, where the \{a_0 \ldots a_{N-1}\} and \{k_0 \ldots k_{N-1}\} parameters represent the weights associated with each metric \{x_0 \ldots x_{N-1}\}.

\[
\text{eMOS} = \sum_{i=0}^{N-1} a_i \cdot x_i^k_i
\]

After subjective testing calibration, the model is shown to provide an adequate performance and closely tracks the users’ perceived QoE.

The proposed approach, while failing to encompass memory effects in QoE such as the ones modeled in [22], succeeds in providing an eMOS model able to encompass the main QoE-impacting factors in HAS (buffer and quality switches characterization plus content quality modeling).

### 2.5.2 QoE Optimization on HAS

Given the broad scope of parameters that affect the final QoE, there are a multitude of aspects that can be optimized in order to increase the overall QoE. The QoE optimization aspects in HAS may be broadly subdivided into three categories (Figure 4):

1. **Content preparation** - Aspects related to how the video is encoded and segmented, e.g.:
Deliverable 3.3

codec; codec parameters; and segment duration;
2. Delivery - Factors intervening in the content delivery process. These may encompass the use of proxies, caches, and network-specific optimizations, to name a few;
3. Consumption - In HAS, client adaptation algorithms play a very significant role in the final QoE. This category encompasses optimizations performed at the client side;

The ensuing sections provide a literature review on these three different categories.

<table>
<thead>
<tr>
<th>PREPARATION</th>
<th>DELIVERY</th>
<th>CONSUMPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>Encoding &amp; Segmentation</td>
<td>Proxy, Cache &amp; Network</td>
<td>Client Adaptation</td>
</tr>
</tbody>
</table>

Figure 4 - QoE optimization aspects on HTTP Adaptive Streaming.

2.5.3 Optimization through content preparation

The content preparation process is crucial on any HAS technology as the process is performed only once but the content is consumed multiple times; thus, any optimization performed at this stage is transparently reflected across the overall HAS solution.

The two dimensions on which optimization is usually performed are encoding and segmentation. In the encoding dimension, different codec parameters may be adjusted such as the codec in use, frame rate, resolution, and Group of Pictures (GOP) size to name a few. As for the segmentation, the segment size dimension plays a significant role on the content buffering and delivery aspects.

The authors of [26] focus on the first aspect of content preparation optimization: encoding. Specifically, on optimizing the encoding process to reduce the streaming bit rate required on adaptive streaming platforms, where the content encoding process should be aware of the segmented nature of the content.

In most commercial H.264 [27] encoding solutions each segment starts with Intra-coded pictures (I-frames), i.e. fully specified pictures, while the other frames are coded with P-frames (predictive-frames). This structure, denoted IPPP, simplifies encoder requirements given that the encoder does not need to be aware of the video content, it just has to generate I-frames at a regular interval. As a side-effect, the I-frame placement is not optimal, and the Rate–distortion (RD) performance is reduced.

To optimize the encoding process, the authors propose a solution based on scene cuts which are used as segment boundaries; thus, the segment size depends on the encoding process instead of being fixed at the typical 2 or 10 seconds. These optimizations are shown to allow for a reduction of about 10% in bandwidth for a given RD target.

In addition to the impact on RD performance, the segment size plays an important role in other aspects of end-to-end HAS solutions. Smaller segments allow for quicker client-adaptation, and lower buffering delays (which may be important in live streaming, for instance), but may present a higher overhead on different parts of HAS technologies, such as the Media Presentation Description (MPD) (or manifest) size - smaller chunks translate into an increase on the total number of chunks, and on larger MPDs being required to describe them.
2.5.4 Optimizing the delivery process

A properly optimized delivery solution is a requirement to ensure that the client-requested content is delivered in a timely and scalable fashion. Given that OTT networks cannot provide any performance guarantees by themselves and span different access technologies (such as radio, Ethernet, and fiber), mechanisms are required to cope with this uncertainty while providing a good QoE to the users.

In [28], the issue of maximizing QoE in LTE wireless networks is considered. Delivering media over wireless links is a well-known challenge, as the shared medium is typically the bottleneck [29] due to its resource availability constraints and high variability. In these networks, each user consuming media with an HAS player will adapt individually to the resources allocated by the scheduler in the eNodeB. However, this scheduler is not content-aware: it merely contemplates channel conditions to perform the scheduling decisions.

As a method for maximizing the overall QoE, a modified DASH server, along with a proxy placed at the base station is suggested. Due to the connection of the proxy server to the eNodeB scheduler, the proxy server is able to gather information on the bandwidth allocated to each client, and performs request rewriting off the client’s segment requests. This approach ensures that a client never requests content with a bit rate larger than the network can handle, avoiding playback stalls, quality up-shifts that are not stable, and allows for an adequate bit rate selection at the beginning of playback, instead of starting with the lowest quality representation. These approaches are shown to improve the QoE MOS by at least 0.4 (in a 1 to 4.5 scale).

While this is an interesting approach, its feasibility may be somewhat limited given that the request-rewriting proxy requires direct information from the eNodeB scheduler.

Another common approach to optimize the delivery process is to focus on caching mechanisms, such as the one depicted in [30], where a QoE-driven cache management system for HAS over wireless networks is presented. The authors focus on optimizing the overall users’ QoE, given a set of playback bitrates, and cache size constraints. Particularly, the cache is populated with chunks from a set of playback rates that are known to maximize the overall QoE (considering every user), and subsequent requests for chunks from the clients are rewritten in order to supply them with the closest representation of the requested bit rate. This provides substantial statistical gains to the caching engine, given that under a particular storage budget a larger number of media items are reused.

Other authors, such as [31] approach the issue from an in-network perspective. The focus of the article is on granting telecom operators a manner of maximizing revenue for on-demand streaming, by allowing the prioritisation of users with higher subscription levels on managed networks. It is intended that “premium” users maintain their ‘premium’ QoE service, at the expense of QoE of other users sharing the same link.

The problem is defined and solved with an Integer Linear Programming (ILP) approach formulated to provide access to all users, while maximizing the client’s utility relative to its subscription level. The results show that, in addition to maximizing the overall utility (w.r.t. to the subscription level), a noticeable side effect is attained: because each client is eventually restricted to a set of tracks so that the overall link capacity is respected, there is a markable reduction in the number of bit rate switches of the clients, which in turn leads to an improvement of user’s QoE.

2.5.5 Optimizing client adaptation mechanisms

The remaining optimization category falls on the client adaptation mechanisms. The client heuristics play a significant role on estimating the adequate chunk that should be requested to the
HAS server. The client must take into consideration factors such as: screen resolution, content frame rate, chunk error rates, bandwidth estimation, and buffer management, to name a few. The interplay of all these heuristics determine the client behavior, and ultimately the user QoE.

Recent studies [32–34] have shown that current commercial implementations of HAS players present several issues in maximizing the usage of the available network resources, as providing a smooth playback, as well as assuring fairness between competing clients.

An excellent evaluation of optimal adaptation trajectories is performed in [35]. The authors create an analytical model of the optimization issue as a Multiple-Choice Nested Knapsack Problem (MCNKP) problem, which is then used as a baseline comparison for the performance of client adaptation algorithms.

In the evaluation section, it is shown that their implementation of a DASH plugin [36] for VLC [37], based on a Proportional-Integral-Derivative (PID)-like controller is able to significantly outperform Microsoft's Smooth Streaming player in a number of benchmarks, such as: rebuffering time, average bit rate, buffer size requirements, and number of quality switches. With regard to fairness in the presence of multiple clients sharing the same bottleneck link, the solutions are shown to perform similarly.

Jiang et al. [34] approaches the client adaptation issue from 3 main perspectives: Fairness, so that competing clients sharing a bottleneck link get a fair share of network resources; Efficiency, to utilize the most out of the available resources; and Stability to prevent unnecessary bit rate switches that may affect the users' QoE.

The issue of chunk download scheduling is carefully analyzed in order to demonstrate that current players relying on periodic download schedulers fail to provide accurate bandwidth estimation in the presence of multiple players. Due to the synchronization between the different download schedulers, different players will observe different bandwidths, leading to unfair shares of the available bandwidth. In order to address the synchronization issue, a randomized scheduler is proposed that relies on the buffer state (instead of solely time) as a basis for scheduling chunk downloads.

Stability in the playback session represents another issue in current players, which are prone to quality switches due to bandwidth variations. Given that stability affects the trade-off between efficiency and fairness, a delayed update method is proposed which may delay bit rate switches if there have already been a certain amount of switches in the recent history.

A more robust approach for bandwidth estimation is also suggested, which, besides considering averages across a set of previous samples, relies on the harmonic mean, instead of the traditional arithmetic mean which is often (mistakenly) used to compute average rates. This approach leads to a more robust average, less susceptible to outliers.

The extensive evaluation performed is able to clearly demonstrate that FESTIVE [34] outperforms the most common commercial implementation of HAS players, such as Microsoft's Smooth Streaming, Netflix's Smooth Streaming, and Adobe's players.
3 Content Distribution

3.1 Typical Content Delivery Network Architecture

CDNs are composed of multiple servers, sometimes called the replica or surrogate servers that acquire data from the origin servers, which are the actual source of the data. The replica servers are interconnected and store copies of the origin servers' content so that they can serve content themselves, reducing the load of the origins.

The issue of how to build a CDN is usually solved using overlay or network approaches, although hybrid solutions are common [38]. The overlay approach establishes application-specific connections and caches on top of the existing network infrastructure and creates virtual interconnections between the network's nodes. In this approach, the network elements such as switches and routers do not play an active part in the content delivery and management process, other than providing basic connectivity and any agreed-upon QoS Service Level Agreements (SLAs) for the CDN nodes. This is a widely used approach in commercial CDN such as Akamai [39] and Limelight [40], as the independence from the underlying network components warrants flexibility not only in terms of the services that can be provided, but also in node deployment.

In contrast, the network approach relies on network elements such as switches, load balancers, and routers to forward requests to local caches and/or application servers according to previously established policies. This is a less flexible form of CDNs that is usually heavily optimized for serving specific content types, and that is often used as a complement to overlay approaches in server farms, i.e. a server farm may internally use a network-based approach for CDN despite being part of a larger overlay network.

![Figure 5 - Typical Content Delivery Network Architecture.](image-url)
Deliverable 3.3

Depending on how the CDN is devised, multiple protocols may be used in the interaction between the different replica servers, such as Cache Array Routing Protocol (CARP) [41], or HyperText Caching Protocol (HTCP) [42]. Apart from these common protocols, each vendor / designer of CDNs usually implements its own communication or interaction protocols, such as Railgun from CloudFlare [43].

In order to get the most performance out of a CDN, it is usual to create platforms that are either tailored or adaptable according to the content they are serving. A good example is the multimedia streaming services of Akamai HD [39], which are optimized for streaming, and the application services of CloudFlare [43], which are optimized for serving dynamic applications.

No single CDN solution is able to address every possible service in an optimized manner without adaptation, i.e., a single optimal and universal solution for CDNs does not exist.

A CDN is in itself a complex network that is expected to perform a multitude of tasks, which are usually subdivided into three main functional components: delivery and management; request routing; and performance measurement, billing and accounting. Apart from billing and accounting, each responsibility will be individually considered and scrutinized. Figure 5 provides an all-encompassing conceptual perspective of the main components of a CDN.

An ideal OTT multimedia CDN is able to deliver content without significant delay and scale-out in order to grow capacity as needed with the simple addition of servers. A state-of-the-art global CDNs is required to sustain bandwidths in the order of Tbps.

3.2 Content Distribution Architectures

In order to understand which content distribution architectures are scalable and viable in a large scale OTT delivery infrastructure, an evaluation must be performed on the possible ones, to identify each solutions' strengths and weaknesses [44, 45].

3.2.1 Centralized Content Delivery

The centralized approach to OTT delivery is the simplest one, where the clients directly target the origin servers without any intermediate edge server, as depicted on Figure 6, and a unicast stream is created directly between a given origin server and a consumer device.

![Figure 6 - Centralized OTT Delivery.](image)

Because consumers target the origin servers directly, this approach provides the lowest delivery delay when streaming live content, and may be cost effective for a few users - small being defined by the maximum number of users that the origin cluster is able to serve simultaneously.
However, this fully centralized approach presents several issues. Firstly, there are security constraints, usually imposed by content providers which forbid users from having direct access to origin servers.

Secondly, this approach does not scale properly with geographically distributed consumers, as users that are further away from the origin cluster will experience increased access delay to the content, which is even more problematic if the streaming session is being conducted using TCP, as this protocol is known to underutilize links when faced with long network delays, especially in high-bandwidth networks - i.e. Long Fat Networks (LFN) [46]. These characteristics will naturally lead to user frustration and reduced QoE.

Thirdly, a centralized approach, without carefully planned content replication requires a large amount of core network and transit traffic, which is particularly expensive in multimedia streaming scenarios, which is known to have high bandwidth requirements.

### 3.2.2 Proxy-Caching

An alternative approach to the centralized solution is the proxy cache architecture, where intermediate, or proxy, nodes communicate directly with the consumers, acquire the content from the origin servers on their behalf, and cache it. This architecture is illustrated on Figure 7.

![Proxy Cache OTT Delivery](image)

This approach presents several benefits when compared with the centralized one. Content security is increased due to the indirect access to the origin server, which may be put in a private network, as long as proxy caches have access to it. Scalability is also increased, by means of caching and geographical distribution of proxy caches, which also provides the added benefit of improvements on users’ QoE due to reduced access latency to the servers in the likely event of a cache hit. Bandwidth costs are reduced through savings in core and transit network traffic.

In spite of these advantages, a proxy cache solution has potential drawbacks, due to two main factors: increased management and deployment complexity; and increased end-to-end delay in the event of a cache miss, which may have a significant impact in the case of live content streaming. However, the benefits of this approach often outweigh its drawbacks.
3.2.3 Peer-to-Peer (P2P)

A P2P approach to distributing OTT content is another possibility where both the proxy caches and the consumers may communicate with each other to locate and acquire content, as exhibited in Figure 8. P2P takes advantage of the uplink capacity of users’ and proxy caches’ connections to lessen the bandwidth burden on the origin servers. An example of a widely used fully P2P streaming service is Popcorn Time [47].

![Figure 8 - P2P OTT Delivery.](image)

In spite of its advantages in terms of utilizing the available upstream bandwidth, P2P streaming presents several challenges that prevent it from being widely deployed on OTT content delivery networks. One of the issues has to do with the startup delay of new streaming session, as locating and acquiring data from peers takes longer than streaming directly from an origin server or proxy cache. Another issue has to do with playback lag in live streaming, as the P2P approach leads to additional delays.

There are also issues with traffic engineering, given that P2P protocols have not been designed to be ISP-friendly. Finally, as with other highly distributed systems, the complexity of deploying, configuring, and managing a fully distributed content delivery network is high, which may be a deterrent for subscription-based streaming service providers that have a responsibility of providing a service with predictable quality to their clients.

3.2.4 Hybrid Delivery

Another possible architecture for an OTT streaming service relies on hybrid delivery, and combines P2P at the clients with the previously discussed proxy cache approach. The diagram of this solution is presented in Figure 9.
Comparing this approach with the full P2P one, several advantages are apparent. Firstly, because the proxy caches may be used to stream content directly to the consumers, like in the simple proxy-cache approach, the additional startup delay caused by the P2P overhead may be mitigated. Secondly, due to peering, bandwidth and load are saved on the proxy caches, thus allowing the solution to better handle flash crowd events, or very popular content, which will be more likely to be available at peers.

The low latency requirement for live content may prevent P2P approaches to live streaming; however, on-demand content may benefit from it.

The downsides are similar to those of full P2P approaches, in that extra complexity is required in the implementation, deployment and management of a hybrid delivery network. Additionally, some terminals, such as Set-Top-Boxes (STB), may not support P2P at all.

### 3.2.5 N-Tier Caching

A design decision that has a high impact on the performance of a CDN is the amount and disposition of caching layers, as well as the storage space available at each layer [48].

The simplest approach to caching within an OTT CDN is to place a single proxy-caching layer at the edge servers, which is responsible for fetching content on behalf of the client and storing local copies according to predefined caching policies, as shown in Figure 7.
In more elaborate approaches, it is possible to add supplementary caching layers, which take the name of aggregation caches, as opposed to the client-facing edge caches. Figure 10 depicts an example of a 2-tier caching solution.

There are several advantages of having aggregation caches on top of the edge caches. When a user moves from one edge cache to the other (as a result of mobile base station change for example), content previously cached on the edge cache and also on the aggregation cache, does not need to be re-requested from the origin server, and may be served directly from the aggregation cache to the new edge cache.

In the event of an edge cache failure, due to a server failure for instance, the aggregation cache is also useful. Thus, given that, in order to rebuild the cache of backup edge caches, there is no need to target the origin server since provided content is present at the aggregation layer.

An aggregation cache is also useful when a CDN has a high geographical diversity and potentially large access delays to the origin server. Using as an example the USA and its 50 states, an aggregation cache could be placed on each state, while edge caches would be installed on each main city.

In these scenarios, a trade-off that must be considered is the cost of adding aggregation caches instead of investing in larger edge caches. Finally, due to the introduction of an additional element in the distribution chain, the end-to-end delay is expected to increase slightly for non-cached items.

### 3.2.6 Content Delivery and Management System

The content and delivery management system is at the core of any CDN. Its responsibilities encompass the replica servers’ physical and virtual placement, content selection and actual delivery, the caching organization techniques, and content outsourcing, i.e., how the content is acquired into the CDN replicas.

#### 3.2.6.1 Replica Server Placement

As the replica servers hold the actual data of a CDN system, their physical and virtual placement plays a very important role on the overall performance of the CDN. Their placement must be carefully planned so that they can be as close as possible to the clients. They usually do not need to be very close to the origin Web servers, as there are typically no significant link bottlenecks between server farms and the Internet; however, if far enough, the network latency might impact their performance.

From a physical location standpoint, the issue of replica server placement may be thought as an instance of the more general problem of placing N servers in M possible locations, such as the facility location of minimum k-median problems [49] or k-Hierarchically well-Separated Trees (k-HST) [50].

These theoretical approaches to replica server placement define a center placement problem using graph theory, where the goal is to identify N cluster centers relying on some measure of “distance” between the centers and the nodes of the graph, i.e., the clients. In the context of CDNs this distance may combine factors such as available bandwidth, delay, or even costs for transit links.

Similarly to k-means clustering [51], cluster cells are created as an outcome of the placement algorithm. This problem is known to be Non-deterministic Polynomial-time (NP) hard; hence, alternative heuristics-based node placement algorithms are commonly used. Examples include Greedy replica placement [52] and Topology-informed placement strategies [53] that leverage existing information regarding the CDNs such as workload patterns and network topology to provide good-enough solutions at a much lower computational cost. The Greedy algorithm [52] is iterative and chooses the location, at each step, that minimizes the cost for clients, out of the available ones. As for
Hot Spot [52], a rank is made regarding the load generated by clients in the vicinity of a possible location, and top N locations are chosen as “hotspots” and consequently the target locations for the N servers. Other derivations of greedy algorithms exist and are discussed in [54].

Some authors proposed dynamic replica placement algorithms [55] that take into consideration QoS requirement for the CDN as well as the maximum capacity of each server/server-farm location; therefore, providing a better approximation of theoretical models. Work in [56] provides a good evaluation methodology on the performance of several heuristic algorithms according to the specific requirements of a given CDN.

Apart from node placement algorithms, there is also the issue of how many replica servers to deploy. This question will vary greatly with the deployment scenario, that may either be confined to a single-ISP or multiple-ISPs [57].

In single-ISP deployment situations, the size and span of the ISP usually limits the number and locations where the replica servers may be placed, and the choice usually falls on major cities and relies on large capacity servers or server farms. This has the main drawback of potentially not having servers nearby clients that need them. If the ISP has a global coverage, this problem is somewhat mitigated.

An alternative is to use multiple ISPs that can provide a broader choice for node placement, and number of nodes that can be placed. This is the approach followed by large international CDNs such as Akamai that have thousands of servers [39,58].

Deciding on how many ISPs to use and how many servers to deploy has an impact on the cost and complexity of the CDN. Deployments with an excessive amount of servers may exhibit both poor resource utilization and lackluster performance [59], whereas using fewer than needed servers will also have a significant impact on the performance, albeit due to the excessively high load. A balance must be struck between the expected utilization of resources and the number of replica servers [60].

### 3.3 Content Delivery Network Interconnection (CDNI)

CDNs provide numerous benefits in large-scale content delivery (as mentioned in previous subsections). As a result, it is desirable that a given item of content can be delivered to the consumer regardless of consumer’s location or attachment network. This creates a need for interconnecting standalone CDNs, so they can interoperate and collectively behave as a single delivery infrastructure [61],[62].

Typically, ISPs operate over multiple areas and use independent CDNs. If these individual CDNs were interconnected, the capacity of their services could be expanded without the CDNs themselves being extended. An example of CDNI between two different CDN providers located in two different countries is illustrated in Figure 11.
As can be seen, CDNI enables CDN A to deliver content held within country A to consumers in country B (and vice versa) by forming a business alliance with CDN B. Both can benefit with the possibility of expanding their service without extra investment in their own networks. One of the advantages of CDNI for content service providers is that it enables them to increase their number of clients without forming business alliances with multiple CDN providers [63].

Another use case of CDNI is to support mobility [63] (Figure 12). Telecommunication operators want to give users the choice of accessing the content and services they want, and pay for, in a wide range of client devices, instead of being constrained to a location (home) or device (TV). As a consequence, with CDNI, users can access the same content seen at home when outside the home on their smartphones or tablets.

As described above, CDNI provides benefits to CDN service providers as well as to consumers, since they can expand their service and the content is delivered, to the end-users, from a nearby replica server, improving QoE. However, since legacy CDNs were implemented using proprietary technology, there has not been any support for an open interface for connecting with other CDNs. Even though several CDNs could be interconnected, it is still difficult to know the consumption of each user in order to charge. Hopefully, Content Delivery Networks interconnection - Work Group (CDNi-WG) is working to allow the CDN collaboration under different administrations [64]. The documents can be accessed online at [65].
3.4 Mobile Content Delivery Networks (mCDNs)

In the last few years, with the advances of broadband wireless technology that provides high-speed access over a wide area, the proliferation of mobile devices, such as smartphones and tablets, is rapidly growing and expected to increase significantly in the coming years. In fact, the increasing number of wireless devices that are accessing mobile networks worldwide is one of the primary contributors to global mobile traffic growth [66].

As a consequence, and because of the constraints and impairments associated with wireless video delivery that are significantly more severe than those associated with wire line, there is the need to consider it as a potential performance bottleneck in next generation delivery of high-bandwidth OTT multimedia content. Three of the primary issues that must be considered for wireless video distribution are:

1. Mobile Device CPU, screen and battery limitations: mobile devices are typically multi-use communication devices that have different processors, display sizes, resolutions and batteries. Usually, mobile devices with smaller form factors utilize processors with lower capability and screen size than tablets for example, but the battery lasts longer. For these reasons, it is necessary that the video assets be transcoded into formats and "gear set" that ensure both good quality and efficient use of resources for transmission and decoding. The battery consumption for video delivery to client device must be minimized.

2. Wireless channel throughput impairments and constraints: the uncontrolled growth of Wi-Fi networks (that is, probably, the most popular technology to support personal wireless networks), particularly in densely populated areas where Wi-Fi networks must coexist, leads to interference problems. The impact of adjacent channel interference depends on the transmission power of the interferers, their spectrum transmit masks, on the reception filters of the stations receiving the interfering signal, and on the statistical properties of the interferers' utilization of the spectrum. This problem can dramatically affect the performance of wireless networks, particularly in live video delivery. Another aspect is the lower available bandwidth and typically smaller displays on mobile devices, the "gear sets" used for wireless are very different than those for wire line. An example is a device with a 640x480 display that can work with video bitrates anywhere between 0.16 and 1.0 Mbps, as opposed to 2.5 Mbps required by a 1280x720 display.

3. Diverse wireless Radio Access Network (RAN): wireless networks can vary between commercial GSM, 3G and LTE up to personal networks such as Wi-Fi and Bluetooth. Video delivery solutions should be RAN agnostic. Should the client side control the requested video bit rate, differences in RAN performance can be automatically accommodated as long as there are enough “gears” provided to span the required range [67].

In order to improve performance in wireless video delivery (with impact in the global optimization of OTT multimedia delivery) it arises the concept of mobile Content Delivery Networks (mCDNs), that is illustrated in Figure 13.
As can be seen, mCDNs are used to optimize content delivery in mobile networks (which have higher latency, higher packet loss and huge variation in download capacity) by adding storage to an AP, such as a wireless router, so that it can serve as a dispersed cache/replica server, providing the content (if it is cached) with low latency and high user experience, preventing network congestion. Although this approach presents improvements, the cache size is reduced compared to traditional proxy caches used in the core network. Thus, the ideal would be to cache only content that would be requested by consumers in the near future.

Consequently, Section 5 will address prefetching and multimedia caching algorithms to better understand how the content in replica servers can be replaced.

### 3.5 CDNs and Multimedia Streaming

As mentioned before, CDNs have been developed to serve the content to Internet users, improving performance and providing scalable services. Although using CDNs brings many benefits, serving multimedia content presents some issues.

To better understand this problem let's take as an example the streaming of high-quality multimedia content. As can be expected, high-quality implies larger size (in MB) than normal multimedia streaming. To keep this content in memory it is necessary a significant amount of free space on the servers HDD. As a direct consequence, this reduces the number of items that a given server may hold and potentially limits the cache hit ratios. Another problem is the significant impact on traffic volume of the network in the event of cache miss, due to its size; it may reduce the user QoE, since it will take more time to populate the replica server. Additionally, if the user does not require all the content or skip portions of a video, many resources are wasted.

Live streaming services represent another challenge to CDN. If the impact on traffic volume is significant, the delay (origin to consumer) is higher and may not be considered anymore “live” streaming service.

In order to tackle these problems, several advanced multimedia stream technologies have been developed, and are presented in the next section.

### 3.6 OTT Multimedia Delivery

The Internet evolved from a simple messaging system [68] built on the shoulders of IP and the end-to-end argument [69] to become one of the most complex systems in operation, both in number of protocols supported and sheer scale.
OTT multimedia networks represent a growing class of media delivery technologies, whose distinctive characteristic is the unmanaged (or open) delivery, without network-supported stream control. The classification of closed or open networks, depends on who controls the network [70].

In a closed (or managed) network, the delivery is performed with the involvement of the ISP, which ensures predetermined QoS features. This is the type of service that is provided on commercial IPTV platforms such as Ericsson’s Mediaroom [71].

On the other hand, in an open and uncontrolled network the delivery takes place without any interference or quality guarantees of the ISPs supporting the delivery, which takes place as if it were any regular Internet content. Because the ISPs’ networks are being used to provide a service from a third-party, which typically uses their network infrastructure for free, this type of delivery is called OTT.

The characteristics of OTT networks enable services to be delivered to the whole Internet, without any capital or operational expenditures on the network infrastructure itself, which are supported by the intermediate ISPs. However, there are some drawbacks to these services: because the networks they rely on to operate are not controlled, no quality guarantees may be ensured, and OTT providers depend entirely on the supporting best-effort network.

This fact raises multiple issues as far as the users’ QoE is concerned. The high-QoE goal requires scalable, reliable, and adaptive services, which must be able to infer the environment conditions in quasi-realtime in order to provide the users’ with the best possible experience at a given point in time. In the context of video delivery, a good experience is correlated with metrics such as low-buffering times, no video freezes or macro-blocks, a video resolution adequate to the viewing-device's screen and, in live events, low end-to-end delay, to name a few.

To frame and provide a good understanding on the services under consideration and how they are delivered, this Section begins by illustrating and discussing each step of the typical content delivery pipeline, and proceeds with a detailed evaluation of a widespread class of OTT multimedia services: that of telecommunication operators, with emphasis on Catch-up TV.

### 3.6.1 Content Delivery Pipeline

As the requirements for bandwidth and demand for ever-richer applications grew, so did the need for systems and architectures able to withstand it in a scalable fashion. In the multimedia delivery context, one that requires huge amounts of bandwidth, a de facto standard delivery pipeline naturally arose and was established to represent the multi-party nature of the multimedia life-cycle, from content creation to consumption.

In order to understand the steps involved in making multimedia content available to users for mass consumption, a content delivery pipeline is exhibited in Figure 14, illustrating the several parts and components of a modern delivery process.

This diagram is by no means exhaustive, but provides a good macro perspective on the delivery process.
3.6.1.1 Content Preparation

The preparation portion of the delivery pipeline performs the actual acquisition of the content in some multimedia format from a live broadcast or Video-on-Demand (VoD) media (DVD, etc), transforms the original content into the content that will be displayed to the user -- e.g. ad insertion in live broadcasts --, encodes it into a consumer-friendly format, such as H.264 or the more modern H.265 codec, and if necessary applies content protection through Digital Rights Management (DRM) mechanisms. At the end of this step, the content is ready to be used as a master source for distribution.

3.6.1.2 Content Ingest

Ingestion is the act of taking a content that is ready for distribution and making it available to the delivery network by placing it in a top level content source. The content is copied into a set (for redundancy and scalability) of so-called Origin servers which hold master copies of the multimedia content and expose it to distribution networks.

3.6.1.3 Content Distribution

The actual content distribution encompasses two main components: the network elements with their associated access technologies (such as Fiber-To-The-Home (FTTH), Digital Subscriber Line (DSL), ...); and the content distribution servers inside the network, that may have different complexity levels.

This component is also responsible for monitoring the provided QoS and QoE, though situations exist where this monitoring is performed by the media servers or the client devices (e.g. in Microsoft Smooth Streaming [72] monitoring is performed by the client as well). The content distribution servers may range from full-blown CDNs, to simple load-balancers ensuring that the available origin servers get similar load shares.
3.6.1.4 Consumption

The final step in the content delivery pipeline is the actual media consumption by the client device. How it is performed depends heavily on the device, the OS, and hardware specifications, such as decoding abilities, screen size, etc. Given the heterogeneity of devices, operating systems, and technologies on the market, this step usually represents a formidable challenge.

3.7 OTT Multimedia Services of Telecommunication Operators

Even though OTT multimedia networks are, by definition, not constrained to telecommunication operators, and key market players exist with “pure-OTT” business models (Google, YouTube, Facebook, ...), telecommunication operators play a very important role in shaping the OTT industry and its services. Their Pay-TV offerings provide rich and interactive services that are widely used across the world; thus, telecommunication operators often provide the push for the massification of new services.

Historically, telecommunication operators have relied on managed networks to deploy their services; however, with the advent of OTT supported services which provide strong competition, they are also moving towards OTT-based distribution systems.

Telecommunication operators may provide OTT services as a multi-screen extension of their Pay-TV services to give users the choice of accessing the content and services they want, and pay for, in a wide range of client devices, instead of being constrained to a location (home) or a device (TV). On the other hand, OTT services enable operators to provide standalone services on otherwise unreachable situations, as the requirement for managed networks is no longer in place, thus broadening consumer choice -- i.e., a given consumer might have Internet from one provider and a linear TV subscription from another.

3.7.1 Linear TV

Linear TV, i.e. “regular TV broadcast” obeying to a predetermined program line-up was considered for decades as the traditional and more popular way of watching TV programs. These were the times where delivery networks were monopolized by public and private operators carrying their own TV programs. Nowadays, this is still the dominant way of watching TV from national free-to-air TV services and major Pay-TV Operators like BT in England; NET in Brazil; Time-Warner in the USA and MEO in Portugal, although customers are moving to other services [73].

3.7.2 Time-shift TV

Time-shift TV relates to the visualization of deferred TV content, i.e., linear-TV content that is recorded to be watched later (from seconds up to several days), using one of the following services:

1. Pause TV is the simplest type of time-shift service, allowing users to pause the television program they are currently watching - from a few seconds to several minutes or even hours. Users can resume the TV broadcast when they want, continuing to watch where they left off; skip a particular segment; or eventually catch up to the linear broadcast.
2. Start-over TV enables users to restart programs that have already started and, eventually, programs that already finished. The amount of time that is possible to rewind varies from operator to operator ranging from some minutes up to 36 hours. The number of TV channels
Deliverable 3.3

supporting this feature is also a decision of the TV operator.

3. *PVR* stands for Personal Video Recorder. In this type of service, the recordings are subject to the user action, i.e., they only occur if the user proactively schedules a TV program or a series to be recorded, or if he decides to start recording a program that is being watched. The behavior of the service is much the same as the one of a VCR (Video Cassette Recorder); however, with a much higher storage capacity and nonlinear access. The user can start watching a recording whenever he wants, even if the program is still being recorded.

4. *Catch-up TV* is the most advanced time-shift service, relying on an automated process of “Live to VoD” [74] (offered by companies like Alcatel-Lucent [75]) or on a more restricted process-based editorial control. With this service, TV operators offer recorded content of the previous days, on a bouquet up to hundreds of TV channels. The time window of the recordings ranges from a couple of hours up to 30 days, and the number of recorded TV channels varies from operator to operator, according to technical, legal, and business constraints. Using this service, users can really, and very easily, catch-up TV programs that have been missed or that they explicitly decided to watch later -- e.g. watching the news after preparing dinner.

3.7.3 Video-on-Demand (VoD)

VoD refers to services where users need to pay to watch a specific content through one of the following ways:

1. Transaction VoD (T-VoD) is the most typical version of the service, where customers need to pay a given amount of money each time they want to watch a content from the VoD catalog. The rental time is usually of 24 or 48 hours, during which the customer may watch the content several times.

2. Electronic Sell Through VoD (EST-VoD) is a version of the VoD service involving the payment of a one-time fee enabling customers to access the purchased content without restrictions, usually on a specific platform. Although this method of VoD is more frequent in OTT providers like Apple iTunes and Amazon Instant Video, it is also being offered by traditional Pay-TV operators, like Verizon’s FiOS TV.

3. Subscription VoD (S-VoD) corresponds to the business model also adopted by OTT providers like Netflix, whereby customers pay a monthly fee that allows them to watch whatever they want from the provider catalog for an unlimited number of times. However, like the EST-VoD version, it is no longer an exclusive option of these providers, since Pay-TV operators are also offering S-VoD. A simple example is the Disney VoD service offered by several Pay-TV operators like AT&T, Cablevision or Comcast.

3.8 Impact of Catch-up TV Services

Several studies indicate a revolution in the television ecosystem due to the introduction of manual and automatic recordings, recommendation and retrieval technologies for television content. Among non-linear IPTV services, Catch-up TV distinguishes itself as the most popular one, even surpassing the popularity of “classical” VoD services such as T-VoDs or EST-VoD [76,77].

Large scale delivery of Catch-up TV represents one of the biggest challenges of Pay-TV, mostly due to two reasons: first, the content must be streamed in unicast to each client, with dedicated
connections per user; second, Catch-up TV content demand is several orders of magnitude larger than that of traditional Video-on-Demand (VoD) content [77].

To lessen the impact of unicast traffic, [78] and [79] suggest the use of decentralized delivery solutions whenever possible, with subsequent studies in [80] for cable television networks, and in [81–83] for IPTV services.

The fact that Catch-up TV is data-intensive is challenging, as it is usually provided as a supplement to Pay-TV subscriptions with no added cost. The network impact of Catch-up TV is expected to keep growing with its popularity, which has been one of the main drivers of an increase in the average time spent by users watching TV [73]. To keep-up with a growing demand, IPTV operators are turning to OTT delivery solutions which do not require investments on managed IPTV infrastructure, and increase the reach of services that may have been previously limited to certain geographic areas.

However, this move requires overcoming several challenges. Given the different requirements of OTT delivery, when compared to that of managed networks, a detailed service understanding is required to properly decide on OTT CDN architectures, plan the physical and logical location of clusters and replica servers, tune caching algorithms, select optimal request routing mechanisms, and estimate computational, network and storage requirements, to name a few.

From an operator's viewpoint, a thorough service comprehension fosters savings on both Capital Expenditures (CAPEX) and Operational Expenditures (OPEX). As an example, CAPEX may be reduced by investing on less extra capacity, because the exact service requirements are known and the delivery system is optimized to meet them, which also contributes to reducing the OPEX.

### 3.9 Usage of Catch-up TV Services

To design OTT Catch-up TV delivery systems capable of operating efficiently, while simultaneously improving users' QoE, it is also essential to understand how the service is effectively used by the clients.

From a behavioral perspective, [84] presents a descriptive and inferential statistical analysis on viewing practices (time-shifted, online and mobile), based on data collected over a six-month period in 2010-2011. The authors consider that the popular time-shift services do not alter the traditional conceptualization of television as a broadcast medium; however, they do not make a clear differentiation between the diverse time-shift services (Pause-TV, Start-over TV, PVR and Catch-up TV). Online viewing, considered an emerging mode that blurs the boundary between television and new media, is seen by the authors as comprising P2P, BitTorrent and video streaming from network TV station sites or dedicated services (e.g. Netflix). As for the motivation that drives respondents to watch content on their computers instead of their TV sets, the reason that stands out is the lack of content availability on broadcast television (42.5%). Finally, they present mobile viewing, mostly through dedicated applications, as the most recent consequence of digital convergence. Despite the potential evolutions registered from 2010-11 until now, the paper gives worthy insights about the differences across three key demographic variables: gender, age, and region of residence.

A recent paper, [85], performs an interesting comparison of broadcast TV viewing behaviors with several nonlinear services (Catch-up TV, VoD streaming services, content recording and downloading). They found that TV series and movies are mostly watched through nonlinear services, and also corroborated that people's attention to content is more focused when nonlinear services are at stake, whereas with regular broadcasts (news, talk shows and other “lighter” television genres) the adoption of multitasking behavior is more frequent. Finally, the authors also illustrate that the hassle of dealing with the several fragmented services, with different qualities, prices, and technological
issues can make it hard for users to watch TV the way they want. This merging of household media
devices and delivery systems was already pointed by Jenkins when he referred to the Black Box
Fallacy [78].

These works are consistent with other research, such as [86], which claims that online content
consumption is more concentrated in time and quantity than offline viewing, contradicting Catch-up
TV's long tail hypothesis. The authors state that 69% of the videos have the same success online and
offline; 16% of the videos are not successful in any platform and only 15% benefit from being
available online. The temporality of replay TV consumption is very close to live broadcasting, thus
softening rather than breaking the synchrony of traditional TV. The largest consumption of online
videos happens in the first 3 days of their appearing, with 58% of the total views. The study is limited
to 11,682 videos available on a 5-month window and to 7 TV channels. Similar results were obtained
by [87], which adds that users overwhelmingly prefer serialized content.

In addition to the research works focused on Catch-up TV, other measurement studies exist that
characterize and model key aspects of IPTV services such as linear/live TV, and T-VoD services. In
the work by Cha et al. [88] the users' live TV channel changing behavior is exhaustively analyzed. The
work's chief conclusions indicate that most channel switching events happen within 10 seconds,
suggesting that users have a very volatile focus. Other key findings pertain to the channels'
popularity, which is found to change with the time of day, and to daily viewing patterns, which vary
with the channels' genres. Gopalakrishnan et al. [89] leverage traces across a 2 year period from a
large-scale IPTV service to provide models for the video request arrival process and stream control
of a T-VoD service. A detailed characterization shows that VoD assets may be grouped into 5 separate
clusters of video lengths, that the video popularity distribution follows an approximate Zipf distribution,
and that a strong popularity drop-off exists as the content ages, showing that a content's recency
influences its popularity.

### 3.10 HTTP Adaptive Streaming

In a scenario with unreliable or varying network conditions, as is the case of today's Internet, which
is a collection of multiple networks all over the world, adaptation plays an important role in improving
the perceived user QoE.

Adaptation is a crucial feature of any streaming technology, as any degradation of the connection
quality (in terms of bandwidth, latency or dropped packets) may cause dropped frames, freezes,
and/or long buffering delays, and render the viewing experience unbearable, especially in the content
being streamed is a live event [90].

There are several ways to provide adaptation, but the most relevant classes are:

- Adaptation as a feature of the content encoding process, i.e. the content is encoded in a
  scalable manner with a baseline quality and with quality improvements if extra data is
  received (Figure 15) - described ahead as the Source Video Coding class;
- Adaptation as a feature of the content distribution process if the content delivered is flexible in
  terms of quality/bit rate, i.e. if multiple representations exist for the same content but with
different quality/bit rate levels (Figure 15) - the dominant class is segmented HTTP-based
delivery.
Hybrids exist that combine these adaptation methods [91]. The specifics of each class and related technologies will be depicted in the subsequent sections.

### 3.10.1 Source Video Coding

Source Video Coding represents a class of video encoding techniques that provide adaptability on the encoding block of the multimedia content creation and delivery pipeline (see Figure 14). These techniques were developed with scalability and robustness in mind, so that the device receiving the content can still properly decode it in situations of varying network quality and availability, even if with a penalty in content quality.

There are currently 2 main scalable source video encoding techniques: Multiple Description Coding (MDC) and Scalable Video Coding (SVC).

#### 3.10.1.1 Multiple Description Coding (MDC)

The concept behind MDC is to generate multiple "descriptions" [92], so that each description contains enough information for the device to playback the content, whose quality improves with the number of descriptions received; hence, scaling in quality.

Media playback quality will be roughly proportional to the total available bandwidth of the content source. Figure 16 shows a sample of the impact of MDC in a case where 2 descriptions exist for a content. Figure 16 a) shows the image restored from both descriptions, while b) and c) demonstrate the result of the individual decoding of each description.
This methodology provides great flexibility in terms of the content source, as multi-homed or P2P networks, for example, are easily supported. In addition, given its resilient nature, the usage of best-effort network connections is a possibility that does not interrupt the playback unless every description is affected.

Several information-theory approaches to MDC exist, and vary on how they use the spatial and temporal information as well as pixel and frequency domain [92,94–96]. Nonetheless, this approach presents some drawbacks. First, there is loss in compression efficiency due to redundancy, but the second and foremost disadvantage is that no standard has been established, which hampers its commercial deployment.

**3.10.2 Scalable Video Coding (SVC)**

SVC is a mature encoding scheme with an industry standard that has been finalized in 2007 (Annex G of the H.264/AVC [27]). The standard promotes a high quality encoding of the original source in a scalable manner and with high encoding efficiency. Scalable, in the context of SVC means that subsets of the original stream may be removed, and the resulting sub-streams can still be decoded by receiving devices, albeit with an impact on the media’s Frames per second (Fps), resolution and/or image quality. It supports format, bit-rate, and power adaptation, along with graceful degradation in lossy transmission environments.

Figure 17 - Scalability modes of Scalable Video Coding (SVC), shows 3 examples on how scalability might be achieved. The first example -- Figure 17 (a) -- demonstrates how temporal scalability might be achieved by adjusting the frame rate; next, on Figure 17 (b) spatial scalability is attained by varying the resolution of the frames; and lastly, on Figure 17 (c) scalability in fidelity is achieved by adjusting quality parameters such as Peak Signal-to-Noise Ratio (PSNR) or compression ratios.

These scalable characteristics make SVC a viable option for targeting different types of devices with the same source content. Mobile phones can use low-resolution / low-fps sub-streams [97] while desktop computers, for instance, may choose to use the complete original stream for the best resolution, frame-rates and image quality. Also, given this scalable nature, the receiving device may adapt on-the-fly to varying transmission conditions [97].

The disadvantage is that while it is based on the H.264/AVC standard, an SVC specific decoder is needed to take advantage of the scalability features, in spite of requiring only a small complexity increase on the decoder for proper support.
3.11 Adaptive Segmented HTTP-based delivery

A more recent approach -- when compared with source video coding -- for delivering content in a scalable manner is to use segmented HTTP-based delivery. The idea behind this method is to encode the original content into streams of different quality, fps, and/or resolution and then fragment those streams into segments - or “chunks” -, usually 2 to 10 seconds long, that can be individually downloaded and decoded (Figure 18).

Segmented HTTP delivery is the natural evolution of progressive download streaming: it provides the same benefits of progressive download without the drawbacks of not supporting adaptation or live streaming, while adding some new features of its own.

By using this approach, the adaptation intelligence is relinquished to the client device, which has to monitor the connection quality and decide which chunk and stream to use at a given point in time. This is a big advantage as many issues generally exist in access networks, and the client is in the best position to assess the network quality [99,100]. Moreover, this inversion-of-control approach enables the client to make decisions not only based on the network quality but also on its own computational resources, which may be limited, for instance in the case of mobile, old or low power devices.

The chunks of each stream are numbered and are time-synchronized with matching chunks in alternative streams; therefore, the client can mix and match chunks from different streams, and the only impact will be on the quality, Fps, or resolution of the media. The client has full control over the downloaded data and may buffer as much, or as less, data as it desires.

This adaptation method has a big advantage over source video encoding methods: because the delivery is independent of the encoding technique, as long as the encoders are able to produce atomic and individual chunks, there is a complete abstraction between the delivery method and the content itself. This decoupling makes the delivery method agnostic to the content and reusable for different encoding schemes. In fact, source video encoded content could also run on segmented HTTP-based delivery [91].

The popularity of this delivery approach grew as multiple vendors implemented their own version of segmented HTTP-based delivery technologies with slightly different characteristics but, in the end, very similar between each other:

- Microsoft specified Smooth Streaming;
- Apple developed HTTP Live Streaming (HLS);
- Adobe created HTTP Dynamic Streaming (HDS);
- MPEG standardized Dynamic Adaptive Streaming over HTTP (DASH) in April 2012 [15].
Quoting Akamai's Will Law [101]: "(the technologies) are 80% the same, yet 100% incompatible. To view HLS, you must have a player for that format. For HDS, another player and for SmoothHD, a third. This fractured delivery space forces encoders, delivery networks and client players to spread their development efforts across all these formats, forgoing optimizations that could be achieved by converging around a single format".

The ensuing sections will discuss HTTP delivery, and detail the specifics of each protocol so that a proper comparison can be made between the competing technologies.

3.11.1 Why HTTP Delivery?

HTTP delivery is at the core of each one of these adaptable technologies, and played an important role in the success they had.

Initial proposals to multimedia streaming had as its main challenges the networks' capacity and delays involved, and lead to the development of Real Time Streaming Protocol (RTSP) [102], a low-overhead streaming protocol with session/state-management features embedded. As the Internet developed, network capacity grew, HTTP became a commodity, and the big challenges in multimedia delivery shifted from the network to the servers' capacity: having servers managing separate streaming sessions for each client is not scalable and makes large multimedia content distribution deployments resource-intensive.

Considering that the Internet was essentially built around HTTP, it has become extremely optimized for this particular method of delivery, where large segments of data are being exchanged. The value of delivering small packets per se, such as TCP packets has diminished, hence the widespread use of progressive download technologies and CDNs to help deal with content locality and reduce the long-haul traffic in the Internet.

A multimedia delivery method using HTTP is then inherently taking advantage of the following facts:

- Most firewalls are already configured to permit HTTP traffic, i.e., TCP port 80, whereas in the case of other streaming protocols this might not be the case – easy firewall and NAT transversal;
- HTTP is known as being a stateless protocol; thus, the streaming session can be managed by the client instead of the server, relieving precious server-side computational resources. Each segment requested will require an individual, short-lived session;
- Reliability and deployment simplicity: HTTP and TCP are widely tested and supported.

3.11.1.1 Microsoft Smooth Streaming

Smooth Streaming [72] is Microsoft's take on adaptive segment-based HTTP Streaming. It builds on the concept of fragmented MPEG-4 standard [103], supports H.264/VC-1 as video codecs, and Windows Media Audio (WMA)/Advanced Audio Coding (AAC) as audio codecs.

It is essentially a proprietary solution, despite Microsoft's efforts to standardize it through Protected Interoperable File Format (PIFF) [104], and its active involvement in 3GPP, MPEG and Digital Entertainment Content Ecosystem (DECE).

The technology has three main components:

- The encoder (usually Microsoft Expression Encoder, though other vendors provide compatible solutions, such as Envivio);
- A Microsoft's IIS Media Services extension which provides the streaming services to the clients' (third party companies also provide compatible streaming services, such as Wowza);
The Smooth Streaming Client: Microsoft provides client implementations based on Silverlight that can be used on Microsoft platforms, as well as client porting SDK, which have been used by third-parties to provide client implementations for popular devices such as the ones running Apple's iOS and Google's Android.

This general architecture is conceptually identical for every segmented HTTP-based streaming protocol. The encoder generates PIFF compliant content (which contains the media itself) as well as two additional files, called "Manifests", which are Extensible Markup Language XML – formatted files. There are two main types of manifests: client and server.

Server manifests provide a very high level perspective on the characteristics of the media file, and are used by the streaming service, usually IIS, as metadata that provides a macro description of the encoded content:
- Number of encoded streams (tracks);
- The track type: video, audio, or text;
- Location, as each track might be stored in a different file — for on-demand content;
- Track content information -- codec type, private data, bit rate, resolution, etc.

There are two variants of server manifests, depending on whether the content is live or on-demand, but the structure is essentially the same. A sample on-demand server manifest is shown in Figure 20.

As for the client manifest, it is must be downloaded by the client and processed in order to initiate the playback. Its data reveals the internal structure of the adaptive content, and provides complete information about the number of available tracks, their encoding, resolution, duration, and how they are fragmented (number of chunks and duration of each chunk -- the default duration is 2 seconds per chunk).

With this information at hand, a client may decide to first request the lowest quality chunks and then, after evaluating the response time of these initial chunks decide whether to scale up the quality of the requested chunks, or not.

A sample (trimmed) client manifest is shown in Figure 21, and a simplified playback flow is shown in Figure 19.

Apart from the adaptive streaming core advantages, Smooth Streaming has additional advantages with respect to extended metadata support, in the form of chapters, markers, subtitles, multiple audio tracks, Digital Video Recorder (DVR) buffer for live content and, most importantly, support for the widely industry-supported Digital Rights Management (DRM) technology Microsoft's PlayReady, which goes one step beyond the typical Initialization Vector (IV) / Advanced Encryption Standard (AES) - based DRM. In a digital world dominated by content providers, this is an essential feature.
Figure 19 - Simplified Smooth Streaming Session Diagram.

```xml
<smil xmlns="http://www.w3.org/2001/SMIL20/Language">
  <head>
    <meta name="clientManifestRelativePath" content="BigBuckBunny.ismc" />
  </head>
  <body>
    <switch>
      <video src="BigBuckBunny_2962.ismv" systemBitrate="2962000">
        <param name="trackID" value="2" valuetype="data" />
        <param name="trackName" value="video" valuetype="data" />
      </video>
      <video src="BigBuckBunny_2056.ismv" systemBitrate="2056000">
        <param name="trackID" value="2" valuetype="data" />
        <param name="trackName" value="video" valuetype="data" />
      </video>
      <audio src="BigBuckBunny_2962.ismv" systemBitrate="160000">
        <param name="trackID" value="1" valuetype="data" />
        <param name="trackName" value="audio" valuetype="data" />
      </audio>
    </switch>
  </body>
</smil>
```

Figure 20 - Sample Smooth Streaming Server Manifest.
3.11.1.2 Apple HTTP Live Streaming (HLS)

HLS is an Internet Engineering Task Force (IETF) Draft [105] that shares many similarities with Microsoft's Smooth Streaming; however, some minor differences exist: first, it only supports H.264/AAC encoding, and requires encapsulation using MPEG-2 Transport Stream (MPEG-2 TS) instead of PIFF fragmented MPEG-4; second, the structure of media metadata is described through a hierarchical use of "m3u8" play-lists.

A top level "playlist-file" (Figure 22) describes the existing tracks -- "media-segments" -- (Figure 23) in terms of content type and bit rate, and also specifies the content encryption, if any. Each media-segment has the pertinent information regarding each media track, such as each segment duration (usually 10 seconds) and additional information in tags.

Technology-specifics aside, HLS is the only adaptive streaming protocol supported by default on Apple devices; hence, it is suitable for content delivery in these environments.

The fact that it uses a different encapsulation from that on from Microsoft's Smooth Streaming is in most cases only a nuisance, given that the encoding method is H.264 / AAC which is also supported.
on Smooth Streaming. Microsoft even allows for real-time / on-the-fly repackaging of Smooth Streaming live streams into HLS on its IIS streaming server.

The main disadvantages of HLS have to do with lack of proper DRM support, as it only supports simple Advanced Encryption Standard (AES) content encryption, the lack of additional experience enriching metadata, such as markers, chapters and so on, and a generalized lack of compatible players, although a few third party ones exist for Linux / Windows environments [37].

### 3.11.1.3 Adobe HTTP Dynamic Streaming (HDS)

Adobe developed its own container for segmented media, F4V, which holds H.264 / AAC encoded content in chunks, based on the ISO/IEC 14496-12 MPEG-4 Part 12 standard [103]. The full specification was defined by Adobe and it is not standardized, although a public document exists describing the media files' internal structure [106].

This streaming protocol also relies on: manifests describing the existing tracks - the F4M [106]; index files, to identify the position of a segment within a stream – F4X's; and segments - F4F's.

The only officially supported media player is Adobe's own Flash Player or players based on Adobe Integrated Runtime (AIR) technology, which limits the number of supported devices, albeit media servers like Wowza [107] are able to take advantage of the fact that the inner content is encoded in H.264/AAC to repackaging it on the fly and support Smooth Streaming or Apple HLS from content based on HDS, though usually at the expense of DRM.

### 3.11.1.4 MPEG Dynamic Adaptive Streaming over HTTP (DASH)

Having covered the main competing technologies in the Adaptive Streaming segment-based HTTP delivery field, one last technology must be mentioned: MPEG-DASH.

This MPEG standard (ISO/IEC 23009-1:2012 [15]) was created to deal with issues associated with vendor-centric solutions, such as the ones enumerated so far. Creating an industry standard, without requiring vendor-specific ecosystems, enables the content distributors to focus more on the content itself and less on technological peculiarities and interoperability issues that rise on fragmented ecosystems, which is paramount, especially if we take into consideration that the vast majority of traffic in the Internet is video [108]: the video encoding, storage, distribution, and playback process must be streamlined and universally supported.

The companies that initially specified their own HTTP adaptive streaming protocols also realized these requirements, as the list of partners that have contributed to the MPEG-DASH specification includes, but is not limited to, Microsoft, Apple, and Adobe.

These reasons also led 3GPP to add support for DASH on its Release 10 specification [109], with further improvements on Release 11 [110]. In recent years, support for DASH has been extended to HTML5, under Media Source Extensions (MSE), and Encrypted Media Extensions (EME). Figure 24 illustrates the evolution of adaptive streaming formats and standards over time.
DASH uses Media Presentation Descriptions (MPDs) to describe time *Periods* (defined by a start time and duration) whose purpose is to facilitate the insertion of different media sequentially, so that scenarios like ad-insertion are possible.

Each time period holds information regarding *Adaptation Sets*, which in turn contain segment information. The adaptation sets contain the encoded alternatives (*Representations*) of a media component, while the segments represent the actual media data. The adaptation sets may contain any media data in its segments, as the technology is video/audio codec agnostic. In addition to the two types of recommended containers (MPEG-4 and MPEG-2 TS), new formats relying on H.265/HEVC are also supported. Figure 25 provides an overview of the hierarchical layers of a DASH Media Presentation Description (MPD).

MPEG-DASH provides an extensive list of features that draws the best from the preceding technologies while adding some new features of its own, such as:

- Seamless advertisement insertion for both live and on-demand content;
- Stream switch, for multi-camera view, multiple audio languages, 3D, and, naturally, multiple bitrates;
- Fragmented MPD, for composing MPDs using multiple sources;
Deliverable 3.3

- Alternate URLs, that allow a client to choose the best suiting content source, which is useful in the context of geolocation optimization, CDNs, or simply load-balancing;
- Support for SVC and MDC;
- Versatile set of descriptors, that can include content metadata, such as content rating, accessibility features and audio channel configuration;
- Support for quality metrics, so that the client may report predefined key metrics to the server;
- Segments of varying duration;
- Clock drift-control for live streaming;
- Flexible DRM support (different DRMs in the same MPD, pay-per-quality, …).

As far as DRM support is concerned, MPEG-DASH is designed to support multiple DRM and Common Encryption (ISO/IEC 23001-7 [111]). Multiple DRM is supported given that each adaptation set may use a DRM scheme independently of other adaptation sets in the same MPD, and as long as the client devices supports one of the specified DRM technologies, it will be able to decode the content.

As far as widespread adoption and research focus is concerned, the fact that this is an open standard, already with significant research initiatives, and that the DASH Industry Forum (DASH-IF) has as its members most of the world's top technology companies, the potential for MPEG-DASH to become the de facto HTTP based segmented streaming technology is significant.

3.12 Content Cache OTT Delivery: Multicast and Unicast Solutions

Multimedia content delivery is a center-piece in today's technological and always-connected society. The Internet's massification, democratization, and ever increasing access bandwidths created excellent conditions for OTT delivery of multimedia content services as can be seen by the huge success of Youtube, Hulu, Netflix, and LoveFilm, to name a few. As noted recently, many Broadcasters are looking to shift up their digital satellite/terrestrial television content to support OTT delivery to as many customers as possible through state-of-the art delivery solutions. This creates new challenges in using different technologies combined, and also in terms of monetization of their investments, according to the strong OTT applications and services demand by the customer's. There are many solutions and architectures that support OTT delivery systems, such the case of accenture Video Solution (AVS), Velocix content delivery, NanoCDN, Cicero's, Cisco's' Videoscape, Edgeware's, Envivio/Verimatrix's, Octoshape's and Pace's to name a few. These solutions will be described in the ensuing subsections.

3.12.1 Accenture Video

Accenture [112] (see Figure 26) through Accenture Video Solution (AVS) delivers digital video services for a widely set of platforms and client devices. The main idea is to monetize the digital video services in an easy-to-use experience, operating widely in a range of unmanaged Over-the-top TV (OTTV) devices, such as set-top boxes, Internet-enabled TV's, tablets, smartphones, PCs and gaming consoles. This solution enables the operation to be performed in managed IPTV set-top-boxes and also live OTTV environments. Basically, this is a multi-channel delivery architecture enabling Telco's to extend their existing IPTV platforms; Cable operators to deliver a common video to multi-screens exposing video-on-demand (VoD) libraries to IP devices. Additionally, the platform is IPTV and OTTV convergent, enabling integration with legacy systems. In terms of network services delivered the AVS offers a best-in-class video quality experience. The applications supported for the AVS are native apps and a full HTML5 application framework to enable multi-screen consumer experience.
3.12.2 Velocix content delivery

Alcatel Lucent [113] (see Figure 27) through Velocix content delivery solution seeks to turn service providers into advanced content delivery platforms with its Velocix CDN architecture. This architecture uses a hierarchical structure composed basically by service, storage and delivery tiers. The Service tier provides the management and control layer for the CDN, and it consists of service nodes and session managers. Moreover, the Service tier also provides a web-based console interface, management controls over a variety of key CDN functions, including routing, geo-configuration and authentication; The Storage tier combines techniques that ingest or register content in the CDN. It also includes Enhanced Origin, Publishing and Storage Appliances as part of the video content objects storage and delivery. The Delivery tier includes three types of caching appliances: Edge Delivery, Intermediary Delivery and Transparent Caches. Edge and Intermediary nodes are deployed hierarchically as well as transparent caches through multicast which delivers live OTT video content, generated basically from popular user-generated content and online video sites.

3.12.3 NanoCDN video delivery

Broadpeak [114] (see Figure 28) shows that multicast still remains a suitable technology to enable OTT delivery through its nanoCDN solution. NanoCDN is a video delivery solution which relies on the
usage of home network equipment as element of the CDN. The idea beyond NanoCDN is to reduce investments by reusing equipment that already exists in the network. NanoCDN is composed mainly by two different components, Transcaster Server and Broadpeak Application: the Transcaster Server converts unicast live ABR stream into multicast, then supplying the end-user viewers through unicast sections. Through Broadpeak Application installed on customer equipment (home gateways) is possible to analyze the traffic going from specific websites and, after that, to redirect it to a cache server. Additionally, one of the many segments of the NanoCDN solution is to provide transparent video caching techniques (VTC). Broadpeak application can work with the CDN mediator server sending usage statistics, mostly in terms of popular video content, and also enabling what type of video content is most popular to be cached on the cached servers.

![NanoCDN Live MultiScreen Architecture](image)

**Figure 28 - Broadpeak’s NanoCDN solution architecture.**

### 3.12.4 Cicero’s content delivery

Cicero [115] (see Figure 29) has developed a solution to deploy an OTT VoIP service. This solution includes Cicero’s OTT client and provisioning server as well as it also includes interfaces with the operators existing systems, such as SIP platform, billing/accounting and CRM/web management. Basically, three components make part of the Cicero’s solution architecture: (i) Cicero Supra, (ii) Cicero Provisioning Server (CPS) and, (iii) Cicero Professional Services. The first one is a VoIP client which runs on android and IOS platforms, and can be customized according to the operator's requirements. The CPS enables the possibility to provision, manage and update softphone client configuration settings. CPS also covers standard customizations required both at client and server side, enabling commercial services such as custom interfaces for authentication, and testing within operator networks.
3.12.5 Ciscos' videoscape

Cisco [116] (see Figure 30) through Videoscape solution addresses the integration of services and OTT video delivery as a single solution. Thus, offering a combination of cloud platform, intelligent network, and client capabilities in order to provide a complete video distribution suite. This suite works through an intelligent CDN platform for a personalized OTT media delivery for multiple screens. For optimized content caching, the Cisco Content Delivery System (CDS), provides video content ingestion, caching, and distribution. Cisco through Content Adaptation Engine (CAE) enables service providers to optimize unmanaged OTT content, caching up popular content and also improving user experience for their subscribers. The main idea is to optimize and monetize the unmanaged OTT content, providing also service differentiation in terms of premium content available on-demand for end user customers.
**3.12.6 Edgeware's distributed video delivery network**

Edgeware [117] (see Figure 31) through Distributed Video Delivery Network (D-VDN) solution delivers content from video from Head-Ends layers and OTT origin servers directly to the access devices clients. The D-VDN solution comprises basically of two commercial products: the Orbit Delivery Server and the Convoy Management Software. Through dedicated hardware the D-VDN Orbit server is able to delivery OTT, Live TV or VOD content to the most popular formats for Set Top Boxes, PCs, Mobile, Tablets, Connected TVs and gaming consoles. Therefore, a management software tool called Convoy provides a collection of streaming statistics into real-time analytics reports for both the content provider and the network operator. This architecture is distributed and integrates video delivery server functions including calculating asset popularity, load and logging adaptive streaming statistics.

![Figure 31 - Edgeware's solution architecture.](image)

**3.12.7 Envivio/Verimatrix’s video delivery**

Envivio/Verimatrix [118] (see Figure 32) joined forces to offer a complete OTT Content Delivery solution, fulfilling the requirements to address Content Protection and Rights Management for OTT content delivery. This solution consists of Envivio4caster encoders in combination with Verimatrix Video Content Authority System (VCAS), performing both security and usage content rights management. Envivio 4Manager Network Management System (NMS) supplies the necessity to management and monitoring of multimedia traffic injected in the network. The Head-End includes VCAS for Internet TV and the Envivio 3-Screen Head-End, thus, performing a multi-format solution for OTT pay TV services to multiple screens. This solution provides a secure OTT content delivery for authorized clients, and also can support differentiated services, as the case of premium OTT pay TV services.
3.12.8 Octoshape’s streaming media delivery system

Octoshape [119] (see Figure 33), by using the multicast technology whenever available and its innovative new streaming media delivery system, outperforms current CDNs solutions, and embraces new trends to perform OTT content delivery as HTTP streaming service. When a video is generated and introduced into the Octoshape system, it breaks the stream up into unique 250kbps streamlets and the streams are coded and replicated in a way that 20 unique streamlets may exist across the Octoshape server, at the end only 4 streamlets are needed to recreate the original 1 Mpbs stream for the user. The transport of those streamlets is provided by UDP (it is compatible with TCP when UDP is not available) with optimized resilience control coding schemes, in order to control and management the stream content delivered to the clients. Additionally, with the introduction of multicast relays at the ISP backbone, it can be used to scale content delivery to many clients, through Native Source-Specific Multicast. As a distribution option, a particular client can receive the streamlets from different multicast sources according to the observed connection quality. In cases of packet loss, the cloud platform can be used and a new streamlet is selected to replace a lost connection, thus introducing more reliability in the content delivery system.
3.12.9 Pace's video delivery

Pace [120] (see Figure 34) OTT solution brings broadcast services by delivering Internet video to set-top boxes, connected TVs, tablets, PCs and games consoles through multi-screen service delivery introduced by the Cobalt's service delivery. This solution enables existing infrastructures to complement their linear TV broadcast services to embrace OTT content demands. The key component of this OTT solution is Cobalt's service delivery platform, enabling the operator to ingest metadata to the viewer's on-screen experience and also deliver content even though the operator's existing pay TV infrastructure or directly from the cloud services. Additionally, in terms of management the Elements Oxygen User Interface (UI) allows the operator to take control of the user experience to customize the services from pure broadcast to hybrid OTT services.

3.12.10 Smartlab's solution

Smartlabs [121] (see Figure 35) OTT solution provides and extends the access to live TVs, radio, movies, and through variety of internet multimedia content to personal computers and mobile devices platforms. Through SmartMEDIA Live TV Streamer which supports content preparation, storage, recording, and enables CoD, nPVR, TSTV, PauseLive services specially played by the OTT solution in both open and encrypted TV streams. The middleware SmartTUBE SDP enables support for widely series of access devices and customers clients. The main idea is to delivery OTT content passing by the SmartMEDIA Streamers, which provides also management of the OTT content delivery to the end-user devices.
Figure 35 - Smartlab’s solution architecture.
4 Cache Management and Distributed Cache Solutions

4.1 Caching Algorithms

The caching storage potential depends on several factors, such as user behavior, content popularity and the caching algorithm itself. The next subsection provides an overview of popular caching algorithms, such as First-In-First-Out (FIFO), Least Recently Used (LRU) and Least Frequently Used (LFU), and the Most Popularly Used (MPU).

In order to understand how some cache management policies work, it is preferable to consider a simple example: the reference string (1,2,3,4,1,2,5,1,2,3,4,5) represents the order in which content is requested (different numbers represent different contents), with a cache size of 3 elements [144].

The simplest page-replacement algorithm is a FIFO algorithm. A FIFO replacement algorithm associates with each page the time when that page was brought into memory. When a page must be replaced, the oldest page is chosen. Table 1 - FIFO cache Replacement Policy demonstrates the application of the reference string to a cache employing FIFO, and shows that 3 cache hits are achieved, along with 9 page faults.

<table>
<thead>
<tr>
<th>Iteration</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
</tr>
</thead>
<tbody>
<tr>
<td>Request</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>1</td>
<td>2</td>
<td>5</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>Result</td>
<td>miss</td>
<td>miss</td>
<td>miss</td>
<td>miss</td>
<td>miss</td>
<td>miss</td>
<td>hit</td>
<td>hit</td>
<td>miss</td>
<td>miss</td>
<td>hit</td>
<td></td>
</tr>
<tr>
<td>Page 1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>Page 2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>Page 3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>4</td>
</tr>
</tbody>
</table>

This algorithm is easy to understand and program. However, its performance is not always good, once the FIFO is well known for being vulnerable to the Bélády's anomaly [145].

LRU replacement algorithm uses the recent past as an approximation of the near future, removing the least recently accessed items as needed in order to have enough space to insert a new item. This approach does not suffer from Bélády's anomaly because it belongs to a class of page-replacement algorithms called stack algorithms [144]. Table 2 demonstrates the application of the reference string to a cache employing LRU, and shows that 2 cache hits are achieved, along with 10 pages faults.

<table>
<thead>
<tr>
<th>Iteration</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
</tr>
</thead>
<tbody>
<tr>
<td>Request</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>1</td>
<td>2</td>
<td>5</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>Result</td>
<td>miss</td>
<td>miss</td>
<td>miss</td>
<td>miss</td>
<td>miss</td>
<td>miss</td>
<td>hit</td>
<td>hit</td>
<td>miss</td>
<td>miss</td>
<td>miss</td>
<td></td>
</tr>
<tr>
<td>Page 1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>Page 2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>4</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>Page 3</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>5</td>
</tr>
</tbody>
</table>
LFU is quite similar to LRU. The main difference is that, instead of removing the least recently accessed item, this approach stores the number of accesses to each item and removes items that are least frequently used. The problem with this approach is when a page is heavily used during the initial phase, but then is never used again. Since it was heavily used, it has a large count and remains in memory even though it is no longer requested. LRU is then the strategy most widely used.

In [146] the authors propose a new caching algorithm: MPU, tested in a Catch-up TV scenario. This approach leverages content demand knowledge to make cache replacement decisions based on “priority maps”. These priority maps contain enough information to unequivocally identify Catch-up TV items and their expected number of requests at each point in the future. Thus, the MPU cache eviction policy favors items that have a greater expected priority, in detriment of others with lower expected priorities. The results show that the use of MPU algorithm provides significantly better cache performance metrics, such as cache cost savings and hit-ratio. Using the same reference string and assuming each item with the priority given by [(1 : 4), (2 : 4), (3 : 3), (4 : 1), (5 : 2)], which maps each item to its respective priority and considering higher priorities map to higher expectations that an item will be requested in the future, the Table 3 demonstrates the application of MPU algorithm and shows that 4 cache hits are achieved, along with 8 pages faults.

### Table 3 - MPU cache Replacement Policy.

<table>
<thead>
<tr>
<th>Iteration</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>11</th>
<th>12</th>
</tr>
</thead>
<tbody>
<tr>
<td>Request</td>
<td>miss</td>
<td>miss</td>
<td>miss</td>
<td>miss</td>
<td>hit</td>
<td>hit</td>
<td>miss</td>
<td>hit</td>
<td>hit</td>
<td>miss</td>
<td>miss</td>
<td>miss</td>
</tr>
<tr>
<td>Result</td>
<td>miss</td>
<td>miss</td>
<td>miss</td>
<td>miss</td>
<td>hit</td>
<td>hit</td>
<td>miss</td>
<td>hit</td>
<td>hit</td>
<td>miss</td>
<td>miss</td>
<td>miss</td>
</tr>
<tr>
<td>Page 1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Page 2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>Page 3</td>
<td>3</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>5</td>
<td>5</td>
<td>5</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

As the performance of caching algorithms is highly dependent on the request sequence, a good caching algorithm is one that knows which items need to be hold in cache, in order to maximize the cache hit-ratio.

### 4.2 Distributed cache content

A distributed cache extends the traditional concept of single cache server spanning content on multiple servers which allows to store and to get the content within the distributed infrastructure itself instead of relying on a single cache server. A distributed cache system may obey some replacement policies when the memory cache is full (e.g., the cache policy must choose which items will be deleted to make room for new ones). As an example, LRU is the most used replacement policy due to its compromise in terms of simplicity and performance. Regarding distributed cache solutions, Redis [122], Memcached [123] and Cachelot [124] are the main ones used by companies like Facebook, Twitter, Youtube, Reddit, Orange, etc. These solutions use the concept of in-memory key-value data model so, for each key there is a single value, and the value is the content being accessed through the search of the respective key.
Figure 36 shows that for each key there is an associated value/content, which the content can be a simple text value or fragments of a video. But there are limitations in the size of the chunks to be placed in cache in order to take full advantage of cache memory. The most likely solution to surpass this limitation is to define a maximum value size to be stored in the memory cache, specially in the case of video chunks.

After having introduced the distributed caching and the key-value data concepts, we are now going to focus on Redis, Memcached and Cachelot solutions.

4.2.1 Redis

Redis [122] (see Figure 37) is an open source in-memory data store structure that is used as a database and memory cache too. It is allowed to store eight different data structures, such as, string, hashes, lists, sets, sorted sets, bitmaps, hyperlogs and geospatial indexes. It is written in C, allowing to store data with optional durability.

In order to achieve performance Redis typically keeps the whole data in memory, but it can be configured to synchronize the data and write it to a file system, ensuring persistence. So, if the node fail, just a few seconds of data is lost.

Redis supports replication master to slave only, therefore, a master can replicate the content to any number of slaves. A slave server is exact a copy of the master server. The replication process is non-blocking at the master side, and any slave server can be a master to another slave allowing single-rooted replication tree, as shown in the Figure 37.
Redis may be used as a cache having the following eviction policies:

- allkeys-lru: evict the less recently used keys first.
- volatile-lru: evict the less recently used keys first, but only among keys that have an expire set
- allkeys-random: evict random keys
- volatile-random: evict random keys, but only evict keys with an expire set.
- volatile-ttl: evict only keys with an expire set, evict those that have shorter time to live (TTL) first.

### 4.2.2 Memcached

Memcached [123] is an open source in-memory distributed high performance cache server. Memcached is often used to speed up websites by using the content stored in memory. At first glance, Memcached stores simple key-value pairs, only strings and integers, and it is more efficient than Redis. Therefore, to store more complex data such as the case of arrays or objects, it needs to be serialized first, and therefore un-serialized. Similar to Redis, Memcached is written in C and the data can be stored with optional durability. In consequence of holding data in RAM memory, if the system restarts then data is lost.

Another limitation is on key length that must be at most 250 bytes. With regard the Clients, Memcached may communicate with servers via tcp, but it does not support replication. Memcached has APIs to provide a very large distributed hash table across multiple servers. Therefore, when the Memcached memory is full, data has to be purged through the LRU policy as well.
Figure 38 illustrates two deployment scenarios. First scenario, each server is completely independent, wasting memory and resources and requiring additional effort to keep the cache consistent between nodes. The second scenario stress the advantages to share the same pool of memory, increasing the memory cache as unique logical structure, thus both web servers share the same memory allocated (128MB) across the entire system.

4.2.3 Cachelot

Cachelot [124] is an open source in-memory LRU Memcached-compatible solution which tends to better hardware utilization and performance in comparison to Memcached solution. The Cachelot API works within a fixed amount of memory, no garbage collector, and its memory utilization overhead is approximately 5-7% of the total memory. Besides memory management, Cachelot ensures smooth responsiveness, for both read and write operations. Cachelot works as a consistent cache, returning an error when out of memory, or evicting old items to free space for new ones. The code is written in C++ and, it can be used on platforms with limited resources, like IoT devices or handheld. Cachelot allows to store and access three million items per second (depending on the CPU cache size), supporting bindings from many programming language: JS, Python, Go, Java, Ruby, Erlang, etc.

Some Cachelot characteristics:

- Single-threaded.
- Communication with the servers via TCP, UDP, and Unix sockets.
- Does not supports content replication.
- Memcache-compatible
- Works as consistent cache
5 Content Prefetching

Prefetching is a generic term used to refer to something that was previously loaded into memory before being explicitly requested. Thus, web prefetching is a technique which reduces the user-perceived latency by predicting web objects and storing them in advance, hoping that the prefetched objects are likely to be accessed in the near future [125], [126], [127]. Therefore, a prefetching mechanism needs to be used in conjunction with a caching strategy.

5.1 Prefetch Techniques

Prefetching strategies are diverse and no single strategy has yet been proposed which provides optimal performance, since there will always be a compromise between the hit ratio and bandwidth [128]. Intuitively, to increase the hit ratio, it is necessary to prefetch those objects that are accessed most frequently but, to minimize the bandwidth consumption, it is necessary to select those objects with longer update intervals [127]. To be effective, prefetching must be implemented in such a way that prefetches are timely, useful and introduce little overhead [128]. Downloading data that is never used is of course a waste of resources.

Many studies on the matter have been made over the years. Prefetching algorithms consider as metric the characteristics of the web objects, such as their access frequency (popularity), sizes and lifetimes (update frequency). This led to the proposal of several prefetching algorithms such as prefetch by popularity, lifetime and good fetch, to name a few.

5.1.1 Prefetch by Popularity

Markatos et al. [129] suggested a “Top Ten” criterion for prefetching web objects. Each server maintains access records of all objects it holds and, periodically, calculates a list of the 10 most popular objects and keeps them in memory [127], [130]. The problem with this approach is that it assumes that all users have the same preferences, and, moreover, it does not keep track of users’ history of accesses during the current session [131]. A slight variance of the “Top Ten” approach is to prefetch the m most popular objects from the entire system. Since popular objects are most likely to be required, this approach is expected to achieve the highest hit ratio [127], [130].

5.1.2 Prefetch by Lifetime

The lifetime of an object is the interval between two consecutive modifications of the object. Since prefetching increases system resource requirements, such as server disk and network bandwidth, the latter will probably be the main limiting factor [132]. Thus, since the content is downloaded from the web server whenever the object is updated, in order to reduce the bandwidth consumption, it is natural to choose those objects that are less frequently updated. Prefetch by lifetime will selects m objects with the longest lifetime to replicate in the local cache and, thus, aims to minimize the extra bandwidth consumption [127], [130].

5.1.3 Good Fetch/Threshold

Venkataramani et al. [133] proposed a threshold algorithm that balances the access frequency and update frequency (lifetime), and only fetches objects whose probability of being accessed before
Deliverable 3.3

being updated exceeds a specified threshold. The intuition behind this criterion is that objects with relatively higher access frequencies and longer update intervals are more likely to be prefetched. Thus, this criterion provides a natural way to limit the bandwidth wasted by prefetching, but does not take into account users' profiles.

Although these methods exhibit some efficacy, recent studies reveal that the prefetching algorithms must take into account several factors, such as the session/consumer behavior and the content itself. Andersson et al. [134] investigated the potential of different prefetching and/or caching strategies for different user behaviors in a catch-up TV network. The objective was to reduce the zapping time, i.e., the time from the channel selection to start of playout or even quick reaction to fast forward or rewind, in order to improve the user's perceived QoE. In this study, consumers were divided into two groups, depending on their behavior: zappers and loyals.

Zappers had as main feature fast switching between programs, searching for the desired view, while loyals consume the first selection to the end before requesting another view. With this division, Andersson et al. [134] wanted to be able to better interpret the behavior of their consumers, in order to improve the service and keep the customer satisfied. The main conclusions reached were that zappers are more prone to watch several different streams per session than loyals, although they had shorter sessions. It was also discovered that zapping prone channels exist. A movie channel, for example, is more likely to zapping behavior in comparison with a mixed channel. Therefore, it is interesting to compare the channel's request patterns and not just the consumer behavior.

An analysis regarding prediction of episodes of the same series comprising different hit ratios is also carried out. Loyals present higher probability than zappers to go from episode X to episode X + 1. Zappers, on the other hand, request previous episodes more likely than loyals. Thus, a good prefetching algorithm should be able to adapt to the profiling of both consumers and contents.

In a similar approach, but without division into groups, Nogueira et al. [135] presents a detailed analysis on the characteristics of users' viewings. The results show that Catch-up TV consumption exhibits very high levels of utilization throughout the day, especially on weekends and Mondays. While a higher utilization on the weekends is expected, since consumers tend to have more free time, the service utilization on Mondays is explained as being due to users catching-up on programs that they missed on the weekends. The superstar effect is also notorious. In an universe of 88,308 unique programs, the top 1,000 programs are responsible for approximately 50% of the total program requests. Results also show that most Catch-up TV playbacks occur shortly after the original content airing and users have a preference for mostly General, Kids, Movies and Series content in virtue of not being time dependent. Thus, Sport and News genres quickly become irrelevant after the first two days.

All data referring to the knowledge of the habits of each user, their preferences and even the days of greater affluence to the service will be helpful in the elaboration of an effective and efficient prefetching mechanism.

In [136] it is presented an algorithm that takes into account information collected from the user session in real time. Bonito et al. [136] conceived this prediction system which must be able to adapt itself to changes in a reasonable time. Thus, a set of “prediction machines” is defined, evolved and evaluated through an evolutionary algorithm in order to obtain better prediction performances using information gathered from user sessions. A user session is defined as a sequence of web requests from a given user. Each request is composed by an identifier “GET”, “POST” or “HEAD” request to a web server coming from a host with specified IP address over the HTTP protocol [RFC 2616]. It is considered the user as an anonymous entity, approximately identified by a combination of the source IP address and the client cookie. The same person connecting to the same site at a later time would be identified as a different user. A request from a user is only considered valid if the HTTP response
code is 2xx (success). Thus, the evolutionary algorithm manages a series of user requests, learning from them and trying to find a pattern in order to predict the user’s next action.

The prefetching techniques do not reduce user latency, they only use the time that the network is not being used, trying to predict the user next action, reducing the response time if the prediction is done correctly.

### 5.2 Replica Servers and Prefetching

One of most used techniques in computing is content replication, which involves sharing information so as to ensure consistency between redundant servers. Regarding the CDNs, and after properly placing the replica servers, it is necessary to decide on how content should be replicated to the replica servers. This is commonly known as content outsourcing, and has been the target of vast research.

Traditionally, three main categories for content outsourcing have been established:

- **Cooperative push-based**: content replication based on pre-fetching with cooperation from replica servers;
- **Non-cooperative pull-based**: content replication similar to traditional caches without pre-fetching and cooperation;
- **Cooperative pull-based**: an evolution of the non-cooperative pull-based where the replica servers cooperate with each other in the event of a cache miss;

The cooperative push-based approach aims to proactively prefetch content that is expected to be requested by clients and replicate it to surrogates according to some predefined rules or cost functions. The content is pushed from the origin servers to the CDN replicas in a cooperative manner, and information is maintained regarding what content is on what servers, allowing easier request redirection. It is clear then that this problem shows great similarities to the replica placement problem; hence, being NP-hard [137] and requiring heuristics for feasible solutions. Greedy algorithms have been shown to provide better performance than other heuristics [58,137].

This approach is traditionally not used on commercial networks given that proper content placement algorithms require knowledge about the Web clients and their demands, which is data that is not commonly available for CDN providers [138,139].

Regarding the non-cooperative pull-based approach, this is the simplest form of content placement on the surrogates. If a client requests a content that is not on the surrogate, a cache miss is triggered, and the surrogate fetches the content from an origin web server. There is no explicit coordination, or cooperation, between either the surrogates or the origin web servers, the content replication is purely client driven. In its simplicity lies the key for successful deployments on popular CDNs such as Akamai or Mirror Image. The drawback is the natural lack of optimization in the server selected to serve the request [140].

In the final approach, cooperative pull-based, which is being used in academic networks such as Coral [141], the content is also not prefetched, but upon a cache miss the surrogates cooperate in order to find neighboring servers that can accommodate the request and avoid requests to the origin servers. This approach typically draws concepts and algorithms from P2P technologies such as Distributed Hash Tables (DHT) to foster cooperation between surrogates [142], but may also rely on Domain Name Server (DNS) redirection to point the clients to suitable surrogates.
5.3 Web Servers and Content Caching Technologies

Having detailed the most commonly employed structural CDN architectures, this subsection details commercial implementations of one of their key components -- the web servers -- along with their chief caching features.

The advances in the field of hypertext systems in the mid 80’s contributed as a starting point to the modern web-servers available today. According to Netcraft’s web server survey [147], Apache (37.00%), Microsoft’s Microsoft’s Internet Information Services (IIS) (30.40%) and Nginx (16.65%) are the most popular web servers on the Internet. These web servers are modular in nature and support caching add-ons capable of turning the web servers into powerful caching systems implementing policies able to take into consideration multiple content types, including live or video-on-demand, static content, and dynamic web pages, to name a few.

Popular solutions for proxy-caching modules and systems include Apache Traffic Server (ATS) [148], Microsoft’s IIS with Application Request Routing (ARR) [149], Nginx [150], Varnish [151], and Squid [152]. They may be combined to optimize the cache performance of a delivery system, e.g. Apache web server with an Nginx proxy-cache front-end. Microsoft’s IIS ARR [149] is an extension for the IIS web server supporting rule-based routing, load balancing and distributed caching.

As for open source solutions, ATS [148] is a modular high-performance proxy server originally developed by Inktomi and recently open-sourced by Yahoo!. Just like IIS, it supports reverse and forward proxy, caching solutions, and also request routing, filtering, and load balancing. Squid [152] is an iconic proxy-cache solution, initially released in 1996 as a fork of the Harvest research project [153], that is frequently used in academic works.

Nginx [150] is another popular web server with proxy cache capabilities, originally developed with the goal of addressing the underperformance and scalability issues associated with Apache server. Nginx has a strong focus on high performance and low memory usage when the subject is to serve dynamic HTTP content. With respect to Varnish [151], it is an HTTP accelerator cache solution designed mainly to support content-heavy dynamic web sites focused exclusively on HTTP content. Varnish stores the cached information in virtual memory and leaves the task of deciding which content to cache in charge of the Operating System (OS). In addition, Varnish works by handling each client connection in a separate worker thread; when the limit of active worker is reached; incoming connections are placed in an overflow queue.

Table 4 provides a comparison of features of the evaluated proxy cache solutions.

<table>
<thead>
<tr>
<th>Supported Features</th>
<th>ATS</th>
<th>Nginx</th>
<th>Varnish</th>
<th>IIS with ARR</th>
<th>Squid</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reverse Proxy</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Forward Proxy</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Transparent Proxy</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Plugin APIs</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Cache</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Edge Side Includes (ESI)</td>
<td>Yes</td>
<td>No</td>
<td>Partial</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Internet Cache Protocol (ICP)</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>Secure Sockets Layer (SSL)</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>SPDY</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>

Table 4 - Comparison of Proxy Cache Solutions.
5.4 Embedded Proxy-Cache Solutions

With the advances in the mobile networks and the Internet itself, OTT video servers are growing in popularity. As mentioned before, in OTT networks the video content is delivered over Internet. The most common video delivery uses the HTTP from conventional web servers given the benefits of fast deployment and Network Address Translation (NAT) traversals [154].

Proxy-caching solutions have growing in popularity. This concept allows a proxy server to act as an intermediary between a user and a content provider, making it possible to cache the most important/frequent content, such as files, images and web pages, allowing to share those resources with more users. These solutions should perform well and require few resources in processing, since they will have to be implemented on embedded systems.

Popular proxy-caching solutions for embedded systems, such as OpenWrt, are Nginx, Squid, Polipo, Tinyproxy and Privoxy. Nginx [155] is a free, open-source and high-performance HTTP server with other functions as well, such as reverse proxy, cache capability and SPDY to name a few. It is known for its high performance, stability, simple configuration and low resource consumption.

Squid [156] is frequently used in academic works. It has extensive access controls and makes a great server accelerator. It runs on most available operating systems, including Windows and is licensed under the GNU General Public License (GPL). Some of its main characteristics are: forward proxy, transparent proxy, reverse proxy and cache capability to name a few.

Polipo [157] is a small and fast caching web proxy, although it does not allow reverse proxy feature. Polipo is no longer maintained.

Tinyproxy [158] is a lightweight HTTP/HTTPS proxy daemon for Portable Operating Interface (POSIX) operating systems. It was designed to be fast and small, ideal for use cases where a full featured HTTP proxy is required, but the system resources for a larger proxy are unavailable. It supports reverse proxy feature, but no cache capability.

Privoxy [159] is a non-caching web proxy with advanced filtering capabilities for enhancing privacy, modifying web page data and HTTP headers, controlling access, and removing advertising and other obnoxious Internet junk. Privoxy has a flexible configuration and can be customized to suit individual needs and tastes. It supports reverse proxy feature.

Table 5 provides a comparison of these five proxy caching solutions, but only focuses on the two main relevant supported features: reverse proxy and cache capability.

Table 5 - Comparison of Five Proxy Cache Solutions - Main features.

<table>
<thead>
<tr>
<th>Supported Features</th>
<th>Nginx</th>
<th>Squid</th>
<th>Polipo</th>
<th>Tinyproxy</th>
<th>Privoxy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Reverse Proxy</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>Yes</td>
<td>Yes</td>
</tr>
<tr>
<td>Cache Capability</td>
<td>Yes</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
<td>No</td>
</tr>
</tbody>
</table>
6 Conclusions

This document provides a thorough overview on the issues and technologies that have an impact on the QoE and performance of OTT multimedia services. To demonstrate the complexity of multimedia delivery solutions, a characterization of the content delivery pipeline is performed.

CDN convey a great deal of responsibility in ensuring that relevant multimedia content is put close to users in order to maximize their service quality and address scalability concerns. They fit within the previously mentioned distribution macro block, and their architectures must be carefully designed to support next-generation multimedia delivery solutions in a scalable and efficient manner; thus, they are also a target for improvement which assesses the performance of different approaches, proposes optimizations, and evaluates dynamic provisioning features.

As multimedia content cannot be properly delivered without suitable streaming protocols, an overview of the existing and new protocols is provided, where it is shown that novel HAS algorithms are expected to keep growing in popularity. HAS protocols are radically different from traditional streaming mechanisms, and increase the strain on the underlying delivery systems by requiring additional storage, for the multiple representations, while increasing fragmentation due to breaking content into chunks.

QoE research in the scope of HAS protocols is still incipient, in spite of being a key concern of any service. Its estimation is shown to be very different than on other traditional streaming technologies and it is still an open-issue. Therefore, QoE assessment for HAS protocols is a challenge that is addressed as well.

Caches are a key component of CDN that deserve special attention and are the focus of a dedicated section where a literature review shows that multimedia services are hard to tackle in an efficient and high-QoE approach, demanding additional research in the face of novel HAS streaming protocols.

A complete OTT multimedia delivery infrastructure optimization is not trivial and is full of open-research challenges to enable a next-generation delivery architecture capable of ensuring the performance, scalability and cost-effectiveness that future services will require.

By investing in large infrastructures to store content is expensive and unattractive. The alternative is to take advantage of mobile terminal to be able to store some content and thus increase the size available for caching. Major challenges are expected in the delivery of OTT multimedia content, due to the increasing number of users. This leads to an inevitably improvement in content predicting, prefetching and cooperation among network nodes on the content delivery. At end, an optimized prefetching approach given its particularities in reducing user perceived latency will be very important to enhance the entire CDN delivery.
References


Deliverable 3.3

[85] J. Vanattenhoven, D. Geerts, Broadcast, Video-on-Demand, and Other Ways to Watch Television Content, in: Proceedings of the ACM International Conference on Interactive


Deliverable 3.3
