Chapter 3 - Methodology

3.1 - Introduction

As designated in the title, this chapter will elaborate regarding the research methodology of the literature review. In this section, the researcher outlines the following aspects following the thesis statement of Quality analysis of HTTP Live Streaming bitrate adaptation methods with multi-tier caching servers.

3.2 - Main points:

- 3.2 - The implementation method of the following subsidiaries:
    - 3.2.1 - HLS & DASH video packaging & delivery
    - 3.2.2 - Centralised Storage with HDD & RAM-Disk infrastructure
    - 3.2.3 - Variable Bitrate Configuration for adaptive bitrate response
    - 3.2.4 - Non-variable Bitrate Configuration
    - 3.2.5 - Separate HLS & DASH video repackaging using cache modules
    - 3.2.6 - IP Based load balancing implementation
- 3.3 - The adopted testing methods
- 3.4 - The study research limitations

3.2.1 - HLS & DASH video packaging & delivery

A virtualised backbone was set up to deploy all the servers required for this implementation. A Linux based NGINX service was used for its powerful Http serving capabilities as well as for the high usage and adaptation by various companies and individuals. A default NGINX service, however, does not support Video-On-Demand packaging; thus, the Nginx-Vod-Module was compiled together with the default NGINX dependencies. By doing so, this allows both HTTP video packaging as well as other features like HTTP services and port forwarding, which are both essential for internet accessibility.

Being that this research is focused on HTTP, HLS & DASH were used since they are the two majority winning protocols used in the streaming world. HLS, which was originally developed by Apple Inc., is the first protocol being tested together with bitrate adaptation, different storage implementations, caching modules and IP based load balancing. DASH which was developed by Google Inc. for their ecosystem which includes Android, Chrome as well as Chromecast, DASH was used to compare and contrast the implementation of HLS as well as the additional steps which would make world-wide VOD streaming possible.

The Nginx-Vod-Module allowed the use of both DASH & HLS to source the video file and split them into segments for HTTP broadcast. This allowed media players to open and receive network streams containing HTTP packets from the packaging server and decode the video; however, a simpler delivery method was needed. A website was set up with a generic player which can read both HLS index as well as DASH manifest streams. This allowed for port forwarding thus being accessible over the internet for testing. However, having the client open a stream directly from the main host without any caches and logic in place will be detrimental to performance in large scale implementations due to the nature of large video file sizes.

3.2.2 - Centralised Storage with HDD & RAM-Disk infrastructure

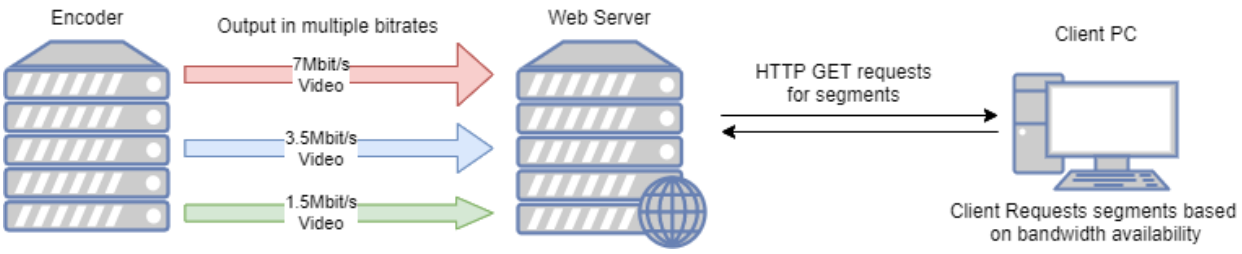
Separate but centralised storage is needed when setting up a VOD service, this is done by setting up a virtualised server with a Linux distro because of its reduced system headroom as well as its high availability for downtime critical applications. By using the Samba software package, two network shares; one for HDD based storage and the second for the RAM-disk based storage,) were created to interface with the packagers.

The HDD based storage was partitioned off from the rest of the system storage. Additionally, the RAM-Disk was created inside the same storage server to avoid significant differences between the servers.

3.2.3 - Variable Bitrate Configuration for adaptive bitrate response

Variable bitrate allows for both the player and the packaging server to adjust the video being served by HLS or DASH to be changed on the fly. This can be done according to the bandwidth available between the client and the origin server. Variable bitrate allowed for the reduction of both the chance for buffering, especially at the very start of initiating the stream, the possibility of dropping frames or the limitation of having to view videos in lower quality for it not to drop frames.

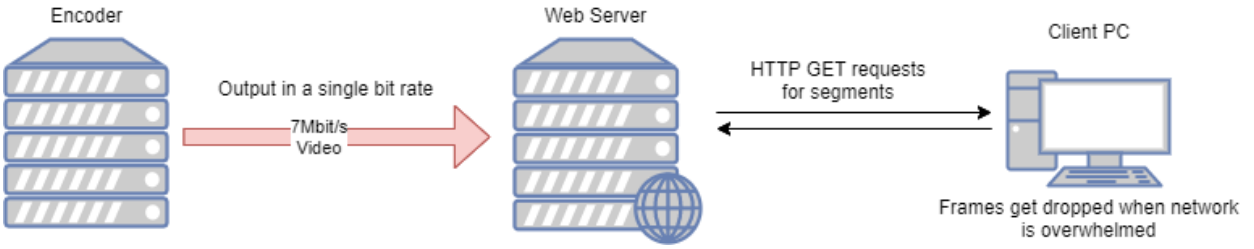
Original video files which are provided by content providers can be made up of 20 Mbit/s bit rate up to 50 Mbit/s bitrate encoding which would be unviewable with standard home internet download speeds. Transcoding video files to be re-encoded in a lower bitrate allows for a better quality-to-file size ratio which would allow VOD providers to both store the files as well as the end client to be able to stream the content without needing large amounts of resources.



All video files used in the implementation were transcoded in three different average bitrates; HD - 7 Mbit/s, SD - 3.5 Mbit/s and SD - 1.5 Mbit/s. This allowed the stream to adapt from SD resolution up to Full HD as well as reducing the storage consumed by the large bitrates. All transcodes where done in H.264 encoding and set all other variables like GOP Size, Level and reference frames to be determined by the original file, in the aim to have all the transcodes as in line with each other as possible.

3.2.4 - Non-variable Bitrate Configuration

To test the implementation, 20Mbit/s and 7Mbit/s video files where also included in the packager without the ability to adapt to the lower bit rate video files. This helped determine the use case of bitrate adaptation when scaling up the sample size of concurrent streams for the server, the client performance and the network performance between them.

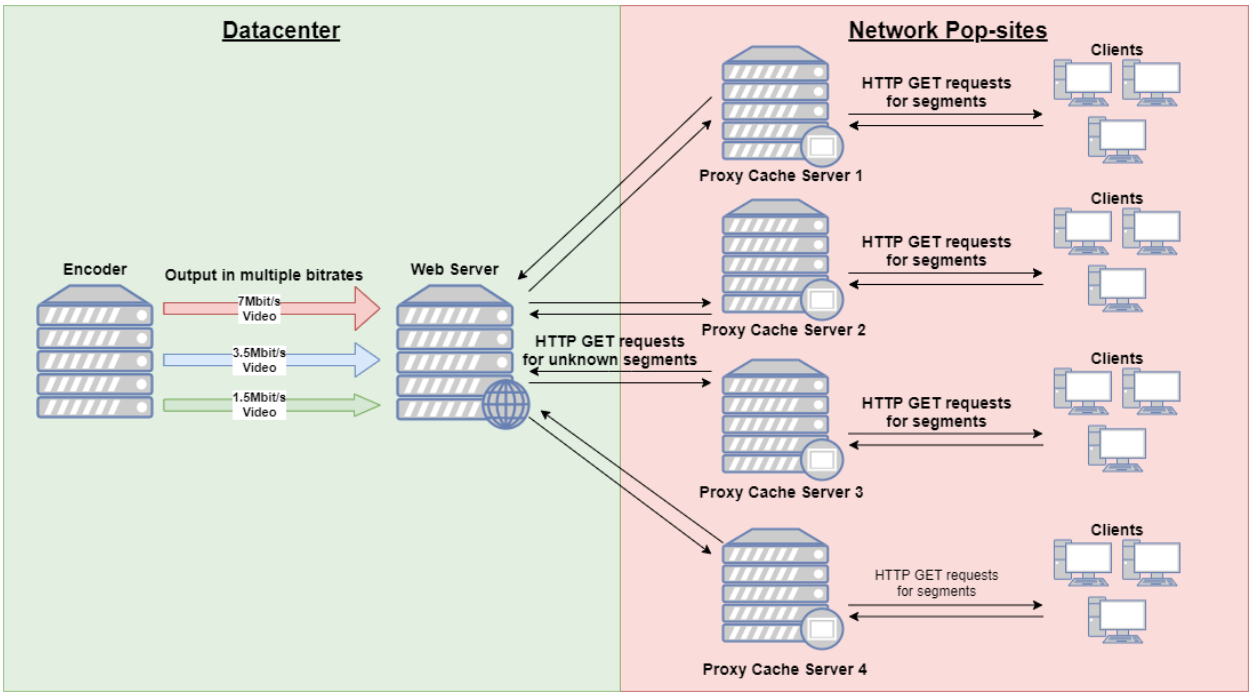


3.2.5 - Separate HLS & DASH video repackaging using cache modules

Cache enabled HLS & DASH video repackaging implementation was achieved by deploying four different Linux based virtual machines, all identical and connected to the main repackaging using the Nginx server features. A new packager was deployed together with network infrastructure to pass the streams through the cache server before being delivered to the end viewer; this was done to enable reliable and more accurate test results.

Cache servers used RAM Disk storage setups, which allowed for a fast response for both the website as well as reduced latency for the stream to be sent. Two dependencies were deployed

to allow both cache functionality through a Proxy-Cache and the proxy forwarding with Proxy-Pass. This allows for the packaging machine to be as close to the stored video files while end-users would be deployed as close as possible to the caching proxies. A central feature of the cache servers is the possibility of storing both metadata as well as manifest files of streams. This means clients which access the cached stream can be served without accessing the main packager. Which results in reduced network traffic on the internal network, furthermore, a popular video will be only transcoded, packaged and sent out the main server once and the cache server would redistribute accordingly.

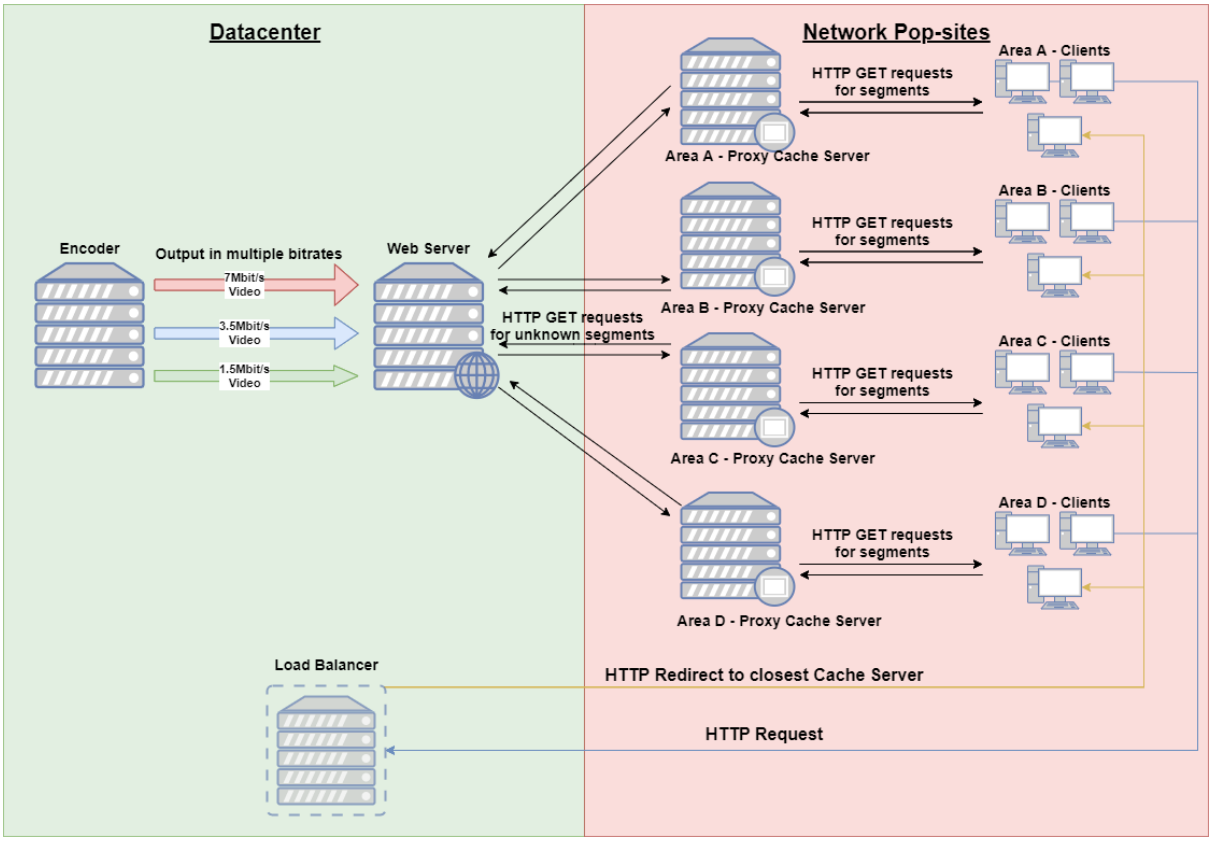


Lastly being that several cache servers can be deployed in various geographical locations, a strategical and progressive roll-out can be achieved by monitoring traffic usage and server capabilities for a phased deployment. This, in the end, allows for smoother roll out for both clients as well as providers.

3.2.6 - IP Based load balancing implementation

Load Balancing was implemented to allow viewers to be assigned to a specific caching server. This is crucial for successfully interfacing with the added caching servers under one domain name or IP address. The load balancer was set up on another separate virtual machine with Linux base and NGINX server. IP Addressing logic was added for four separate networks to interface with four individual cache servers representing a distributed network. A fifth network was set up to interface directly with the main VOD packager, to test and evaluate the traffic management and reduction when using cache servers compared to when connecting solely to the origin server.

To represent the different users who would be accessing the service from different locations, client VMs where deployed, with each client PC connected to one of the individual networks. Additionally, to ensure correct connectivity, as well as correct IP address logic processing by the load balancer, an additional webpage on a different port address, was added with unique identification for each cache server. This allows ensuring correct connectivity while testing the network deployment.



3.3 - The adopted testing methods

Testing methods include the following subsidiaries:

3.3.1 - HLS vs DASH performance comparison

3.3.2 - HDD based Storage vs Cache based storage

3.3.3 - Adaptive Bitrate vs Single Bitrate comparison

3.3.4 – Single Node vs Dispersed Caching Nodes

3.3.5 - Load Balanced Setup vs Point to Point

3.3.1 - HLS vs DASH performance comparison

HLS vs DASH
Server Performance Ratio
Client Performance Ratio
Video Quality
Quality of Experience
Scalability

Performance load testing for each packaging protocol was implemented; each test was carried out with a single client directly connected to the server and pulling a single bitrate file. Additionally, performance counters were taken from the server as well as the client, to determine differences in processing from both sides. Video quality was also taken into consideration to determine the higher quality performer. Lastly, simulated requests were created in order to load the packaging server. Scalability counters were tabulated for data analysis.

3.3.2 - HDD based Storage vs Cache based storage

HDD Storage vs Cache Storage
Reliability
Load Handling Performance
Monetary Feasibility
Scalability

For the following test, the network was bundled to support 2Gbit/s, this ensured that HDD usage would be the first to be impacted. Data was requested by a direct connected high-performance machine with a RAM-Disk. This ensured that bottlenecks were avoided and the Hard disk drive was first to start impacting performance. The test was repeated, this time for the RAM-based storage and results where tabulated. Additionally, other variables based on other research papers were also tabulated, with the aim to obtain results which show how practical and feasible are the two storage methods.

3.3.3 - Adaptive Bitrate vs Single Bitrate comparison

Adaptive Bitrate vs Single Bitrate
Tune in Speed
Buffering Ratio
Network performance
Quality of Experience
Scalability

The testing conducted for Bitrate adaptation included the use of network loading applications like IPerf, together with the implemented system to make the server adjust the bitrate according to the bandwidth available. The tests are run on a standard gigabit ethernet connection, and IPerf

settings were kept identical for each test run. Results were taken for the aforementioned variables. After processing the data gathered, graphs were then exported.

3.3.4 - Single Node vs Dispersed Caching Nodes

Single Node vs Multiple Caching Nodes
Client Handling
Network performance
Quality of Experience
Scalability

The mentioned variables will be populated by running a set of tests which were meant to focus on the server-side of the implementation. This can be done by heavily loading the single server with requests for it to send video files to the clients. When under load, measurements were taken for the number of clients it can handle, network performance, quality of experience (mid-video buffering or long tune in times) and scalability. All tests were run with both HLS & DASH on HDD based storage from the primary storage as well as with a single bitrate for tests to be as constant as possible.

3.3.5 - Load Balanced Setup vs Point to Point

Load Balanced Setup vs Point to Point
Client Allocation Handling
Network performance
Quality of Experience
Scalability

Additional to the Single Node vs Dispersed Caching Nodes test, an additional test was carried out to compare the effectiveness of having a load-balanced configuration to a point-to-point deployment. Once again simulated loading of the servers will take place and measurements will be taken for the following aspects; client allocation handling, network performance from the main packager, QoE and System Scalability.

3.7 - The study research limitations

Due to the nature of this study, simulated network traffic had to be used in order to effectively put the system on a load, meaning that results may slightly vary when attempting to use these systems for production systems.

Additionally, only a few select of videos were ever imported onto the service. Only two variants of files, one at 7 Mbit/s and another which could adapt between 1.5 Mbit/s, 3.5 Mbit/s and 7 Mbit/s could be used so to limit the number of variable changes between one test scenario and another. The majority of testing had to be done with one-gigabit uplinks due to the limited amount of hardware resources available. Live environments would typically have uplinks between 10 gigabit and 100 gigabit, having said that, the implementation focuses on the sample of results gathered for larger implementations to base their system structure on.

Lastly, another process which would allow better and more efficient use of VOD is on-the-fly-transcoding, this would allow the use for only one source video file per title to transcode on-demand, rather than three previously transcoded files to be stored in the NAS. Thus, reducing storage wastage and reducing database complexity. On-the-fly-transcoding was not included in this study because of the limited hardware resources as well as the time needed to build a functioning platform as well as tests to quantify the performance.