MUSLIN demo: High QoE Fair Multi-Source Live Streaming

Simon Da Silva[1], Joachim Bruneau-Queyreix[23], Mathias Lacaud[12],
Daniel Négru[1], Laurent Réveillère[1]
[1] Univ. Bordeaux, LaBRI, UMR 5800, F-33400 Talence, France
[2] Joada SAS, Bordeaux, France, and [3] National Institute of Telecommunications, Warsaw, Poland
{sdasilva,mlacaud,reveillere,negru}@labri.fr,jbruneauqueyreix@joada.net

ABSTRACT

Delivering video content with a high and fairly shared quality of experience is a challenging task in view of the drastic video traffic increase forecasts. Currently, content delivery networks provide numerous servers hosting replicas of the video content, and consuming clients are re-directed to the closest server. Then, the video content is streamed using adaptive streaming solutions. However, some servers become overloaded, and clients may experience a poor or unfairly distributed quality of experience.

In this demonstration, we showcase `Muslin`, a streaming solution supporting a high, fairly shared end-users quality of experience for live streaming. `Muslin` leverages on MS-Stream, a content delivery solution in which a client can simultaneously use several servers. `Muslin` dynamically provisions servers and replicates content into servers, and advertises servers to clients based on real-time delivery conditions. Our demonstration shows that our approach outperforms traditional content delivery schemes enabling to increase the fairness and quality of experience at the user side without requiring a greater underlying content delivery platform.

CCS CONCEPTS

• **Networks**;

KEYWORDS

live streaming, multi-source adaptive streaming, fairness, QoE

ACM Reference Format:
Simon Da Silva[1], Joachim Bruneau-Queyreix[23], Mathias Lacaud[12], Daniel Négru[1], Laurent Réveillère[1] . 2018. `MUSLIN` demo: High QoE Fair Multi-Source Live Streaming . In *MMSys'18: 9th ACM Multimedia Systems Conference, June12–15, 2018, Amsterdam, Netherlands.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3204949.3208108

1 INTRODUCTION

End-users' Quality of Experience (QoE) is a crucial factor for the success of the increasing number of video streaming services. According to Cisco [2], video traffic will experience a tremendous growth and is expected to exceed 80% of the total Internet traffic by 2020. Most of the time, such traffic increase forecasts are not followed by the necessary upgrade of core networks capacity due to the important costs it incurs and major issues arise with respect to the Quality of Experience of such services. Therefore, the design of current and future content delivery solutions needs to consider such aspects.

Content Delivery Networks (CDNs) are extensively used for the delivery of video content over the Internet. In such architectures, geographically distributed replica servers located as close as possible to the consuming clients are provisioned in advance with sufficient capacities using estimates of the expected workload. When accessing a content, consuming clients are automatically re-directed to the closest server so as to temper network congestion and achieve higher throughput. Although CDN solutions can handle a large volume of requests, they laboriously adapt to the highly dynamic and volatile nature of live streaming service audiences. As a consequence, the streaming infrastructure can rapidly be either over-scaled incurring unnecessary expenditures, or under-sized and thus delivering poor QoE to end-users.

In addition to the CDN-based infrastructure, streaming services usually rely on HTTP Adaptive Streaming (HAS) solutions, often relying on the widely adopted *Dynamic Adaptive Streaming over HTTP* (DASH) standard. Such solutions enable the consuming client to dynamically adjust the requested content bitrate according to the observed network conditions or to the client buffer occupancy. However, if a large amount of end-users located under the same geographic area is simultaneously consuming the same streaming service, the nearest server may become rapidly overloaded. As a consequence, some users may suffer throughput degradation or content unavailability, and experience a poor or unfairly shared QoE as they compete for network and server resources.

We introduce `Muslin`, a streaming solution supporting a high, fairly shared end-users quality of experience for live streaming services over the Internet. `Muslin` leverages on MS-Stream, based on the DASH standard, in which a client can simultaneously use several servers with heterogeneous capacities in order to aggregate network throughput on multiple communication channels. To support live streaming, `Muslin` periodically estimates the required throughput to adjust the scale of the service infrastructure. `Muslin` then assigns content servers to clients based on feedbacks (containing QoS metrics and other relevant data) periodically sent by `Muslin` clients during streaming sessions.

MMSys'18, June 12–15, 2018, Amsterdam, Netherlands

ACM ISBN 978-1-4503-5192-8/18/06.
https://doi.org/10.1145/3204949.3208108

2 CONCEPTS: MS-STREAM AND MUSLIN

In this section we present the scientific and engineering concepts behind our contribution.

2.1 MS-Stream

The Multiple-Source Adaptive Streaming over HTTP (MS-Stream) [5] [8] [6] [7] solution is a proposition that extends the DASH standard, wherein a client can simultaneously utilize multiple servers in order to aggregate bandwidth over multiple links while being resilient to network and server impairments. In MS-Stream, for a given video segment, each considered server delivers a video sub-segment to the client.

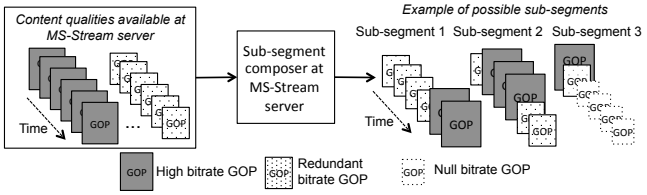

Figure 1: Sub-segment generation and composition

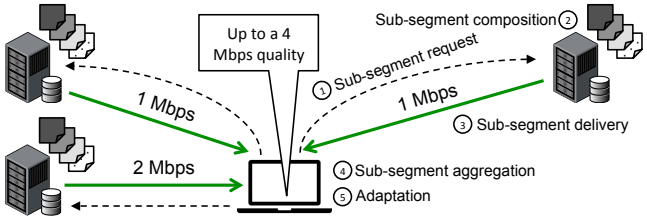

Figure 2: MS-Stream content delivery overview

As shown in Fig.1, sub-segments are generated by interleaving GoPs at different bitrates for the same segment: a high desired bitrate, a critically low bitrate (redundant bitrate), or an emtpy GoP. The redundant bitrate is set to low values (e.g. 150 Kbps) in order to provide video playback at the lowest possible network transfer cost. Reconstructing the original content quality is achieved by selecting the GoPs of higher size in the pool of received sub-segments at client-side. Should some sub-segments be missing, the content is still playable by relying on the redundant GoPs, hence displaying a sub-optimal visual quality but providing reliability and less rebufferings in fluctuating network conditions.

An overview of the MS-Stream functioning is depicted in Fig. 2. A MPD file containing the available MS-Stream servers and video segments is periodically delivered to the client. The client instructs MS-Stream servers to generate and deliver sub-segments composed of video GoPs from the representations available (listed in the MPD file). Then, the MS-Stream client merges the received sub-segments to reconstruct a playable video segment with the highest possible visual quality. The client adapts the number of simultaneously used servers according to the observed network conditions and to the targeted bitrate. The client also attempts to minimize the bandwidth consumption overhead ($O\%$) resulting from GoP redundancy. This redundancy adds about 6.5% network overhead on average. It ought to be noted that the generation and aggregation of sub-segments have very low processing footprints [7] as they only require to assemble already encoded GoPs available at different bitrates. A demonstration of MS-Stream is available online [1].

2.2 MUSLIN: Multi-Source Live Streaming

`Muslin` goal is to provide a high and fairly shared QoE for live streaming services. To do so, it tackles the main reasons why end-users are not satisfied with their streaming experience, which are the number of rebuffering events, considered the main negative impact on perceived QoE [13], the average video bitrate displayed on the user video player and the number of resolution changes during the session, as both have a significant influence on QoE in adaptive streaming [9]. `Muslin` intends to solve the root causes for such QoE degradation, the two main reasons being (1) the server load and (2) the low bandwidth between the server and the client. Indeed, if a server is overloaded or if the network channel bandwidth to this server is low, clients requests to this server will timeout and cause rebufferings or visual quality degradation. Therefore, `Muslin` is able to monitor current delivery conditions to adapt its delivery schemes.

The `Muslin` system is composed of a `Muslin` server, MS-Stream clients, and MS-Stream content delivery servers with a `Muslin` overlay to handle feedbacks and provisioning. Indeed, `Muslin` clients send periodic feedbacks to the `Muslin` server, including the observed bandwidth from each server, the video sub-segment requests failure (timeout) rate, their average displayed video bitrate, the number of rebufferings they experience, and the number of quality changes. Then, based on these feedbacks, the `Muslin` server accordingly scales the underlying delivery platform, re-allocates servers, and re-advertises content servers to `Muslin` clients to provide a better QoE to end-users.

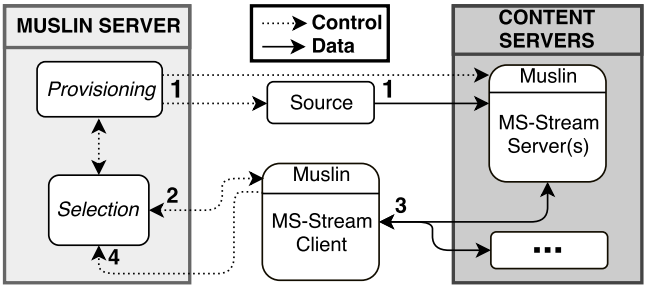

Figure 3: `Muslin` system architecture overview

As illustrated in Fig.3, (**1**) the `Muslin` server dynamically provisions content servers and replicates content to available MS-Stream content delivery servers, which then register themselves to the selection module; (**2**) when a client requests a MPD file, the selection module replies with a list of available servers; (**3**) the client can access live content and begin the streaming session with the MS-Stream protocol;

(4) Muslin clients send periodic feedbacks. In this section, we present in details the Muslin system and the Muslin server two main components, the provisioning module and the selection module.

2.2.1 Provisioning module. The provisioning module goal is to decide on the number of servers to provision not only to answer end-users throughput demand in video contents, but also to maximize their QoE and minimize the required infrastructure scale. To do so, it periodically estimates the required throughput to fulfill the demand based on actual feedbacks, and provisions a subset of servers to host the content. To decide which servers to provision, the Muslin server computes a score for each MS-Stream content server. This score is based on the clients locations, and on feedbacks gathered periodically from all clients. Besides, this provisioning algorithm is run on the Muslin server every T seconds. This period T is equal to the length of two segments (typically 10 seconds).

2.2.2 Selection module. The Muslin selection module goal is to advertise a subset of available content servers to each client, based on a Ranking Score RS_{sc}, in order to reach a high and fairly shared QoE. Then, Muslin clients decide how many servers they use, based on MS-Stream adaptation strategies. As illustrated in Fig. 4, if the closest content server is already overloaded, the Muslin server selects and advertises other content servers with a higher RS_{sc} to the client. It prevents content starvation from clients, and allows fairness among users independently from their geographic position or nearby servers.

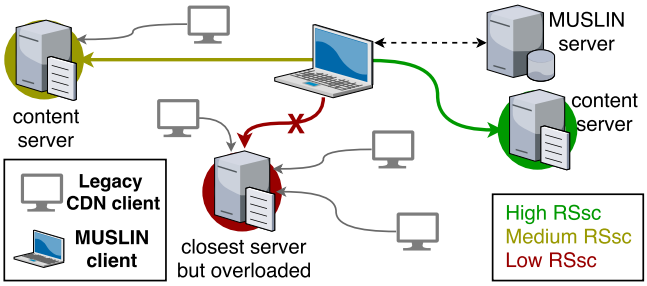

Figure 4: Muslin RS_{sc}-based servers selection example

First, the selection module returns an ordered list of servers when a client requests to discover available content servers. To order the list of servers, the selection module uses a client-specific Ranking Score (labeled RS_{sc}) for each server s and client c, based on feedbacks periodically sent by Muslin clients during streaming sessions. Similarly to the provisioning score, the RS_{sc} is based on the distance between each client and server, and on clients feedbacks. As shown in equation 1, the client-specific ranking score includes the maximum distance between any two places on Earth (20000 kilometers), the geographical distance GD_{sc} using geoIP data inferred from IP addresses, the video sub-segment delivery failure rate FR_s of server s (i.e. the percentage of requests the server was not able to handle on time), and the average observed bandwidth OBW_s between all clients and server s.

$$RS_{sc} = ((20000 - GD_{sc}) * (1 - FR_s) * OBW_s)^{\frac{1}{3}} \quad (1)$$

The selection module computes the client-specific Ranking Score RS_{sc} between each client c and each currently provisioned server s, and returns the MPD file containing servers sorted by descending RS_{sc} order.

2.3 Implementation and scalability discussion

The Muslin modules and Muslin content servers overlay are implemented in Java and run inside light-weight Docker containers. Muslin content servers are built on top of MS-Stream servers by adding the necessary glue code to manage the interaction with the Muslin provisioning and selection modules. All interactions with the Muslin modules fulfill the REST architecture style. Muslin clients are developed in pure JavaScript and run within any mobile or desktop Web browser. Clients extend MS-Stream clients by featuring periodic feedback reports to the Muslin server.

In terms of scalability issues, the Muslin system scales similarly to current HAS solutions as MS-Stream is compliant with the DASH standard. A scalability downside is due to the periodic clients feedbacks as the Muslin server workload grows linearly with the number of clients.

To solve this issue, we implement on the client a feedback request probability Pr to bound the number of feedbacks (see equation 2). We thus ensure statistically that at most N clients will send a feedback for every period T, depending on the current audience v_t.

$$\Pr = \min(1, N/v_t) \quad (2)$$

Another scalability downside is due to the MPD refresh requests from Muslin clients every few segments, or when they experience a poor QoE. Similarly to the clients feedbacks, the Muslin server can become overloaded when too many clients request a new MPD file. To solve this issue, the Muslin selection module is distributed across several network nodes, each node only handling nearby clients requests (routed using classic DNS-based schemes).

3 NOVELTY OF THE WORK

This section shows our innovations beyond current systems.

3.1 Multi-Source HTTP Adaptive Streaming

The work of Adhikari et al. [15] advocates that QoE would greatly benefit from the venue of a practical HAS that can actually utilize multiple servers simultaneously. Whang et al. [12] present a streaming proposal using several servers in parallel, providing better fairness, efficiency and stability at server side, but lacking segment scheduling that leads to low QoE performances against path heterogeneity. A multi-source evolution of DASH is introduced by Pu et al. [16], employing the Scalable Video Coding (SVC) technique that imposes dependency between layers and requires a great care in the design of the layer scheduler to prevent video stalls.

In contrast with these approaches, our client-centric multi-source solution (MS-Stream) is an extension of the DASH standard that considers requesting sub-segments to several servers simultaneously, and abandoning late sub-segment requests. The retrieved sub-segments can contain redundant data so as to be aggregatable and independent from each other, hence exposing a suitable network-impairment and server-failure resiliency mechanism to avoid video stalls due to the volatility of resources in heterogeneous environments.

3.2 Servers selection, QoE and fairness

Although CDN operators keep their strategies secret [17], the usual paradigm is to estimate the audience for an event, and to provision enough servers near end-users to withstand the demand. Then, when clients request video content, the CDN strategy is to route their requests to the nearest server thanks to DNS [4] or IP anycast [3], and use HAS protocols for delivery. This behavior minimizes network-induced latency, and lowers the probability to encounter congestion. For instance, Adhikari et al. [15] introduced the DASH framework of Netflix, the largest DASH provider worldwide, and outlined that a user is always bound to one server, regardless of network issues. Consequently, one major drawback is that servers can get overloaded, and thus some clients may receive a poor QoE or might even not have access to the content at all. Therefore, `Muslin` takes into account not only the distance, but also the server bandwidth and requests failure (timeout) rate, enabling to provide a better QoE to the users.

Besides, there have been some attempts to reach a better QoE fairness between HAS clients. Georgopoulos et al. [10] use Software Defined Networks to allocate network bandwidth, and Petrangeli et al. [11] adapt the requested video bitrate. However, to the best of our knowledge, all approaches towards higher QoE fairness are single-source oriented and do not consider dynamically advertising servers to the clients.

4 DEMO DESCRIPTION AND SETUP

In this demo, the `Muslin` and MS-Stream stack is compared to standard DASH live streaming with anycast server affectation. To do so, a set of servers with controllable bandwidth is provided (see Fig. 5).

Demo scenario. The tester is in the use case of a end-user willing to watch a live stream. He can thus select the available servers and their upload bitrate. Both video players are displayed on the screen, along with several QoE metrics. The tester can thus see in real-time the average video bitrate displayed, the number of rebuffering events, and the number of resolution changes during the session.

Results. By taking into account clients real-time feedbacks, `Muslin` enhances users' QoE and fairness through multi-source streaming with a small network overhead. In previous experiments, we demonstrated an increase of 100 kbps in median displayed bitrate, 2.5 less quality changes per minute, and fewer rebufferings compared to a best-case CDN implementation. We also registered an increase of 19.6% in bitrate

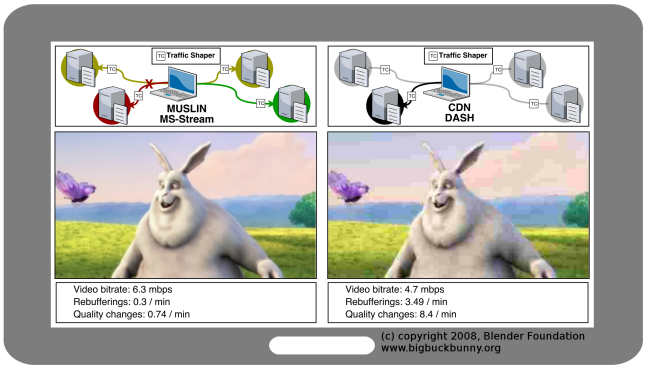

Figure 5: Demo - `Muslin` client (left) selects multiple servers to provide a high QoE and fairness, while the legacy CDN client (right) receives content from a single server

fairness, 52% in quality changes fairness and 23.6% in rebuffering fairness, using the F index described by T. Hoßfeld et al. [14].

REFERENCES

[1] 2017. MS-Stream Demonstration: http://msstream.net. (2017).
[2] Cisco. 2016. VNI. (2016). cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/complete-white-paper-c11-481360.pdf
[3] A. Flavel et al. 2015. Fastroute: A scalable load-aware anycast routing architecture for modern cdns. *connections* 27 (2015).
[4] E. Nygren et al. 2010. The Akamai Network: A Platform for High-performance Internet Applications. *SIGOPS Oper. Syst. Rev.* (2010).
[5] J. Bruneau-Queyreix et al. 2017. A multiple-source adaptive streaming solution enhancing consumer's perceived quality. In *IEEE Consumer Communications and Networking Conference (CCNC), demonstration track*. Las vegas, United States.
[6] J. Bruneau-Queyreix et al. 2017. MS-Stream: A multiple-source adaptive streaming solution enhancing consumer's perceived quality. In *IEEE Consumer Communications and Networking Conference (CCNC)*. Las vegas, United States.
[7] J. Bruneau-Queyreix et al. 2017. QoE Enhancement Through Cost-Effective Adaptation Decision Process for Multiple-Server Streaming over HTTP. In *IEEE International Conference on Multimedia and Expo (ICME)*.
[8] J. Bruneau-Queyreix et al. 2018. Adding a new dimension to HTTP Adaptive Streaming through multiple-source capabilities. In *IEEE Multimedia Magazine*.
[9] M. Seufert et al. 2015. A Survey on Quality of Experience of HTTP Adaptive Streaming. *IEEE Communications Surveys and Tutorials* (2015).
[10] P. Georgopoulos et al. 2013. Towards Network-wide QoE Fairness using OpenFlow-assisted Adaptive Video Streaming. *ACM SIGCOMM Workshop on Future Human-Centric Multimedia Networking* (2013).
[11] S. Petrangeli et al. 2015. QoE-Driven Rate Adaptation Heuristic for Fair Adaptive Video Streaming. *ACM Trans. Multimedia Comput. Commun. Appl.* (2015).
[12] S. Zhang et al. 2015. Presto: Towards fair and efficient HTTP adaptive streaming from multiple servers. *IEEE International Conference on Communications (ICC)* (2015).
[13] T. Hobfeld et al. 2011. Quantification of YouTube QoE via Crowdsourcing. In *IEEE International Symposium on Multimedia*.
[14] T. Hoßfeld et al. 2017. Definition of QoE Fairness in Shared Systems. *IEEE Communications Letters* (2017).
[15] V. K. Adhikari et al. 2012. Unreeling netflix: Understanding and improving multi-CDN delivery. *IEEE INFOCOM* (2012).
[16] W. Pu et al. 2011. Dynamic Adaptive Streaming over HTTP from Multiple Content Distribution Servers. *IEEE Global Telecommunications Conference (GLOBECOM)* (2011).
[17] A. Passarella. 2012. A survey on content-centric technologies for the current Internet: CDN and P2P solutions. *Computer Communications* (2012).